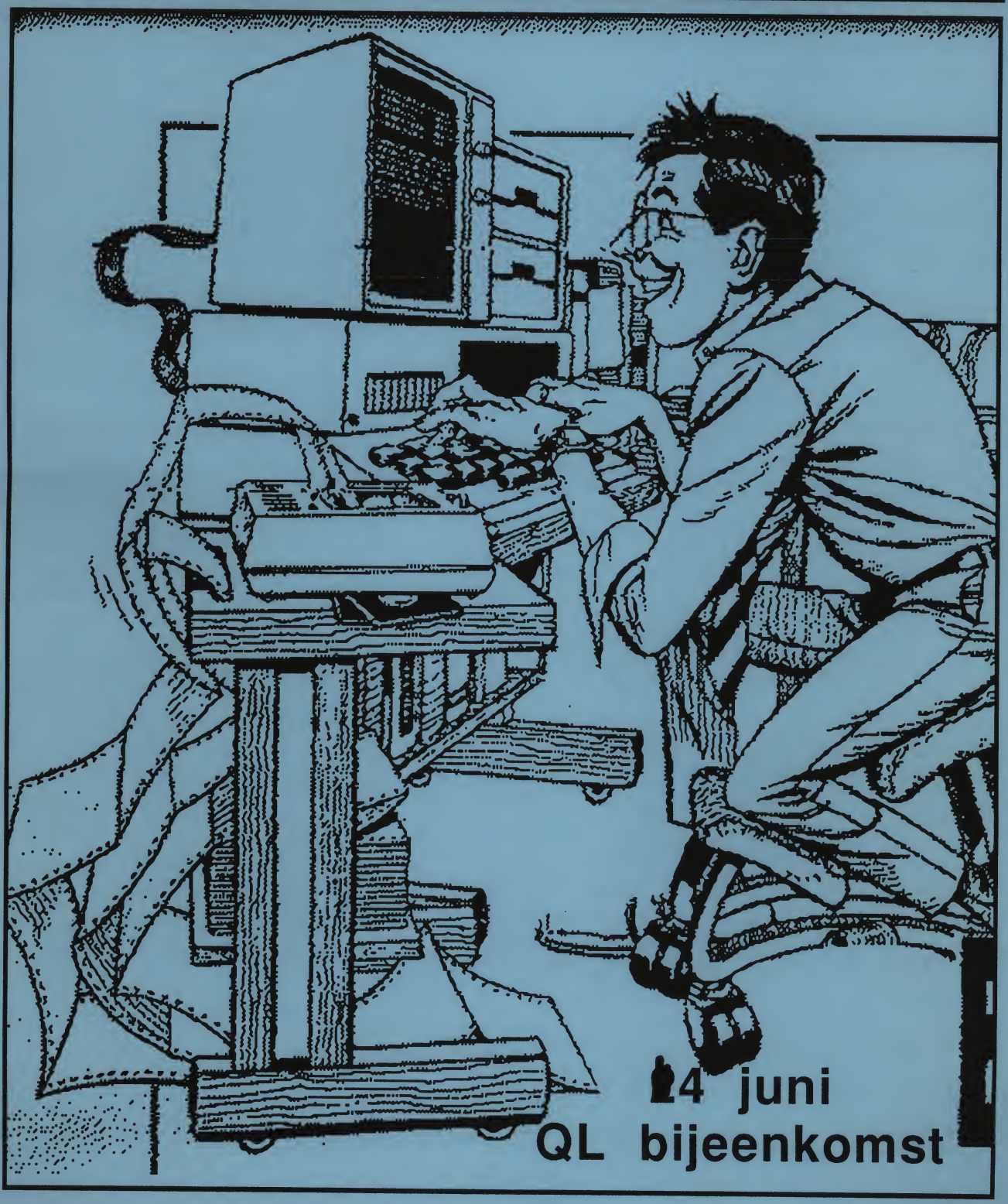
Quasar

Mei 1988 nr.35



4 juni QL-dag in Utrecht

INHOUD QUASAR 35

KOLOFON

Stichting SIN_QL_AIR
Rotterdam
giro: 4597345

ADMINISTATIE
sekretariaat
PENNINGMEESTER
Nabestellen oude nummers.

BOB VISSER
Scheepmakerskade 30
3011 VX Rotterdam
Tel. 010 - 414.3554

VOORZITTER

RON DEN BREEMS
Kroonstaddreef 27
3067 RT Rotterdam
Tel. 010 - 455.1234

REDAKTIE, layout en samenstelling Quasar.

GERARD VAN ROOIJEN
Gruttostraat 15
3435 DJ Nieuwegein
Tel. 03402 - 33027

DATABANK
Tel. 03404 - 22533
Sysop:
MICHEL & WILLEM SPANJER
Hortensialaan 11
3702 VD Zeist
Tel. 03404-20581
SVP alléén tussen 19.00 en 22.00 uur

HARDWARE
Reparaties en onderdelen

MICHEL & WILLEM SPANJER
Hortensialaan 11
3702 VD Zeist
Tel. 03404 - 20581
SVP alléén tussen 19.00 en 22.00 uur

VRAGEN OVER:
Superbasic, Pascal, Machinetaal, Quill, Archive, Abacus, Hardware

KEES V.D. WAL
Kwekerijstraat 22
2613 VE Delft
Tel.015 - 140367
SVP alléén tussen 20.30 en 23.30 uur

VRAGEN OVER:
Machinetaal en Hard-ware

ARD JONKER
Tel. 020 - 230795

VRAGEN OVER:
Machinetaal en C

MARC KOOL
Tel. 020 - 429345

Vanaf nu zal het mijn taak zijn de Quasar maandelijks compleet samen te stellen.
Allereerst wil ik Ron bedanken voor zijn inzet van de afgelopen 34 nummers, en ik ben blij dat ik nog steeds terug kan vallen op zijn opgedane ervaring, want Ron blijft gewoon op de achtergrond aanwezig.

Zoals jullie allen zullen weten valt en staat dit blad geheel met inzendingen van onze lezers, dus blijft het zaak om alle interessante ontwikkelingen op QL-gebied aan ons door te sluizen, zodat wij deze het in het blad op kunnen nemen.

Er is veel belangstelling voor kleine (of grotere) Hardwareprojekten geïllustreerd met een kompleet schema en indien mogelijk met printplaat(tekening) en bouwbeschrijving, ook voor algemene toepassingen in ARCHIVE, EASEL. ABACUS en meer informatie over het gebruik van de programmeertaal C.
Dus heb je zelf wat hardwarematig aan je QL gebroddeld, klim in de pen (en daar gebruiken we tegenwoordig Quill voor) en probeer zo duidelijk mogelijk de zaken op papier te zetten, zodat ook de personen met wat minder ervaring in elektronica deze uitbreiding toch na kunnen bouwen.
Wat ARCHIVE betreft, heb je een interresante toepassing geschreven in ARCHIVE laat ons hem dan publiceren zodat men kan zien wat er zoal mogelijk is in ARCHIVE. (En dat is niet weinig).
De programmeertaal C leent zich uitstekend voor het zelf uitbreiden van procedures en functies, het lijkt ons interessant deze te plaatsen zodat medegebruikers van C hier hun voordeel mee kunnen doen. (Zie artikel elders in dit blad.)
Ook vragen hoe een bepaald probleem in C opgelost zou kunnen worden zijn welkom.

Wij zien uw inzendingen gaarne tegemoet als QUILL-doc of als ASCII-file op cartridge en vanaf nu is het ook mogelijk gebruik te maken van 5 1/4 inch floppy-disk met 40 of 80 tracks (Wel even vermelden s.v.p). Het is voor ons ook erg makkelijk als er uitgeprinte versie bijgesloten wordt.

Met vriendelijke groeten, Gerard

KORTE BERICHTEN

KOMIN FAILLIET

De vermoedens hieromtrend waren in het vorige nummer al ter sprake gekomen, en helaas moeten wij deze geruchten bevestigen.

Het faillisement van Komin is op 30 maart 1988 uitgesproken.
Curator:
Mr. R.J.M. van Bree
Postbus 581
5600 AN Eindhoven
Kantoor houdende aan de:
Fazantlaan 27
Eindhoven
Dus een ieder die nog gelden of materialen van Komin tegoed hebben doen er verstandig aan zich te wenden tot bovenstaande curator.

Is er interresse in 1200 Baud full duplex voor onze Databank

De sysop van onze databank vraagt zich af of er mensen zijn die er belang bij hebben dat de databank ook gaat draaien op 1200 baud full duplex.
Diegene die zelf een modem hebben met de mogelijkheid van 1200 baud full duplex en gebruik maken van de Databank worden verzocht dit aan Michel Spanjer door te geven, hetzij via het bulletinboard of telefonisch.

Telefoonnummer QL Databank
03404 - 22533
(24 uur per dag)

Telefoonnummer Michel Spanjer
03404 - 20581
(van 19.00 - 22.00 uur)

MICROFAIR

Binnenkort is het weer zover,op 11 juni in Londen start al weer de 27ste Microfair. De Microfair is de enigste regelmatige terugkerende beurs waar alles op QL gebied te vinden is en meestal tegen gunstiger tarieven.
Ook dit keer ga ik weer naar de Microfair toe en natuurlijk zal ik deze keer weer mijn best doen om de spullen,die onze leden daar willen bestellen via mij,zo goedkoop mogelijk te krijgen.
Vorige keer was het wat kort dag van de vooraanmelding maar deze keer ben ik er op tijd bij.
Dus als u wat wilt bestellen dan staan hieronder een aantal aanwijzingen.

- Kijk in de QL WORLD wat je wilt hebben, schrijf dit op een briefje met je naam en adres, doe het bedrag wat het kost aan Engelse Ponden erbij en stuur dit naar mij op. (aangetekend of voor eigen risico). (vergeet niet de retour porto kosten bij te sluiten).
- Of neem even kontakt met mij op als dit alles niet geheel duidelijk is.

Een paar prijzen:

- Trump card	£	175,00
- Ace card	£	131,00
- Trey card	£	131,00
- 512k	£	89,00
- Miracle modem	£	45,00
- Schon keybord	£	45,00
- 3x 64k Eprombord	£	40,00

Let wel op deze prijzen zijn in Engelse ponden.

Voor de rest en voor de software gelden de normale prijzen vermeldt in de QL WORLD.

Uiteraard zal ik mijn best doen om zoveel mogelijk korting zien te bedingen.
Ik hoor nog van u allen.

Fred van der Neut
01807-10553 (alleen woensdagavond tot 22.00 uur)

CONTRIBUTIE

Zoals jullie allen gemerkt zullen hebben is er bij iedereen een acceptgirokaart in de bus gerold.
Nu is de betaling van de contributie van groot belang voor het continueren van onze vereniging. Gelukkig hebben reeds vele leden gebruik gemaakt van de acceptgiro voor betaling van hun contributie .
De leden waarvan wij de contributie nog niet ontvangen hebben vonden bij deze Quasar een nieuwe acceptgirokaart bijgesloten. Deze leden vonden ook 2 uitroeptekens op hun adreslabel. Wij verzoeken deze leden vriendelijk alsnog de betaling d.m.v. de acceptgirokaart in orde te maken.
(Het is mogelijk dat uw betaling en de toezending van deze Quasar elkaar gekruist hebben.).
Wat is ƒ 40,- nu, voor een heel jaar informatie over de QL d.m.v. een maandelijks verschijnend informatieblad met daarbij nog diverse QL-bijeenkomsten. DOEN!!!

Donderdag 27 oktober a.s.

Internationale QL-Dag in België

Waar aanwezig zullen zijn afgevaardigden uit: Frankrijk
België
Duitsland
Engeland
en waarschijnlijk nog andere landen.
Details volgen z.s.m.

TWEE KOPPEN EN DUS SCHIZOFREEN

De manier waarop QL-files door disc-interfaces moeten worden gelezen en geschreven is vastgelegd in een standaard, maar het blijkt dat die standaard ofvoor meer dan een uitleg vatbaar is of dat men zich er gewoon niet aan houdt. Al eerder is in Quasar opgemerkt dat de manier waarop door CST- en aanverwante disc-interfaces aangegeven wordt hoeveel ruimte er op de floppy over is, verschilt met die van het "officiele" Sinclair- (Micro Peripherals)interface.

Het laatste telt kennelijk de tabellen na waarin wordt bijgehouden waar elke file staat en hoeveel ruimte zij inneemt, het CST-IF leest en beschrijft een voor het totaal bestemd veld in de eerste sector op kant 0 (begin van het "mapping block").
Op zich nog geen probleem. Ernstiger wordt het als het om gewone code-files gaat die met LBYTES geladen worden en als er EXE-files in het geding zijn.
Voor elke file staat de 64 bytes lange fileheader twee maal op de schijf: een exemplaar in de directory en een aan het begin van elke file. De eerste 16 bytesvan de header bevatten o.a. de volgende informatie:

1. Lengte van de file (inclusief de header);
2. Type file (kan zijn: exe-file, bij JS en later relocatable objectfile en tenslotte andere filetypen);
3. Lengte van dataruimte voor EXE-files (indien van toepassing);
4. Lengte van de filenaam.

Het CST-interface leest en schrijft 1, 2 en 3 alleen in de fileheader die in de directory staat (van 1 schrijft het 64 = 40H in de andere header); het MP-interface schrijft 1 en 4 in beide exemplaren, maar vreemd genoeg 2 en3 alleen in de header aan het begin van de file.
Verder lezen LBYTES en EXEC(_W) via dit laatste interface 1 en 2 (en vermoedelijk ook 3, heb ik niet naar gekeken) alleen in laatstgenoemd header-exemplaar (LBYTES heeft alleen 1 nodig).

Gevolgen voor een disc die door een CST-interface (het meest courant geloof ik) is beschreven en via een MP-interface wordt gelezen:

a. met LBYTES laadt een MP-interface nul bytes in het geheugen (leest nl. 64+nul in het veld voor de file-lengte);
b. met EXEC(_W) krijg je de melding "bad parameter", omdat in het veld voor het filetype iets anders staat dan QDOS op grond van het gegeven commando verwacht.

Remedie:

- voor probleem a:
gewoon de file kopieren; het COPY-commando leest de file-lengte in de directory en zet de lengte van de file ook in de header aan het begin van de file;

- voor probleem b:
dat is wat lastiger. Een mogelijkheid is het uitspitten van de directory naar de file in kwestie met de bij het MP-interface meegeleverde sector-editor. In de 16 bytes voor de eigenlijke filenaam staan alle gegevens die je nodig hebt. Maak een kopie van de file; laad deze in het geheugen met RESPR en LBYTES en save het geheel met sexec op een andere disc of cartridge, waarbij de lengte van de code en de hoeveelheid benodigde dataruimte opgegeven moeten worden die in de fileheader in de directory staat. Een tweede aanpak is het kopieren naar een verder lege disc en het met de hand bijwerken van de fileheader aan het begin van de file, na eerst als boven de directory van de oorspronkelijke disc te hebben doorzocht.
DIE DISC MOET JE DUS NIET WISSEN, aangezien je dan gegevens van de EXE-files weggooit.

Het is in principe doenlijk om de headers van de EXE-files op de disc zelf aan te vullen met de informatie die in de headers in de directory staan. Niet geheel ongevaarlijk, en je moet met enig beleid naar het begin van elke file zoeken, tenzij je de tabellen in het "mapping block" kunt ontrafelen.
Het gemakkelijkste is natuurlijk om alleen kopieen te draaien via microdrive.

Nog wat gegevens:

1. structuur eerste deel van fileheader (high bytes steeds het eerst):

-4 bytes file-lengte;
-1 byte file access: niet gebruikt;
-1 byte filetype: 1 indien EXE-file, 2 voor relocatables (soms), 0 voor andere filetypen, zoals SB-programma's;
-4 bytes dataruimte voor EXE-files of eventueel (maar vermoedelijk nooit gebruikt) extra gegevens voor andere filetypen);
-4 bytes ongebruikt;
-2 bytes lengte filenaam (rijkelijk op timistisch: voor de filenaam zelf staan maar 36 bytes te beschikking. Is dit gedaan voor een mogelijkheid tot copy-protection ?).

Te vinden in de Sinclair QDOS Companion (Pennell) pag. 50, en de Sinclair Technical Guide, pag. 38.

2. directory-structuur:

files staan op track 0 in de volgorde 110, 410, 710, 200, 500, 800, 210 enz. (sector, kant, track; sectoren geteld van 1-9).
De eerste eigenlijke file op een diskette begint op 601.

Er is gewerkt met een MP-interface versie 5.3E, alleen met dubbelzijdig geformatte discs.

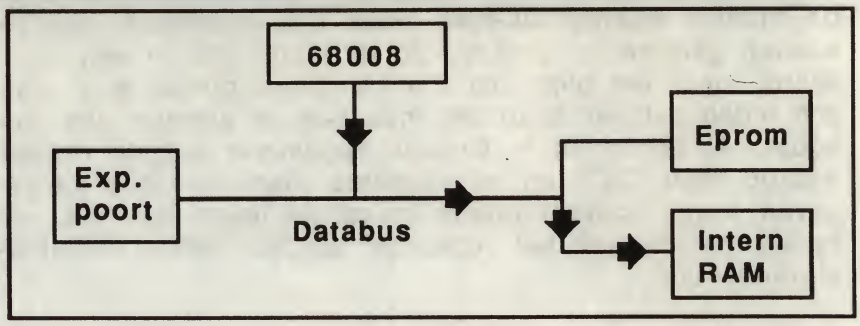
Wiard Smid

BUFFER VOOR UITBREIDINGSPOORT

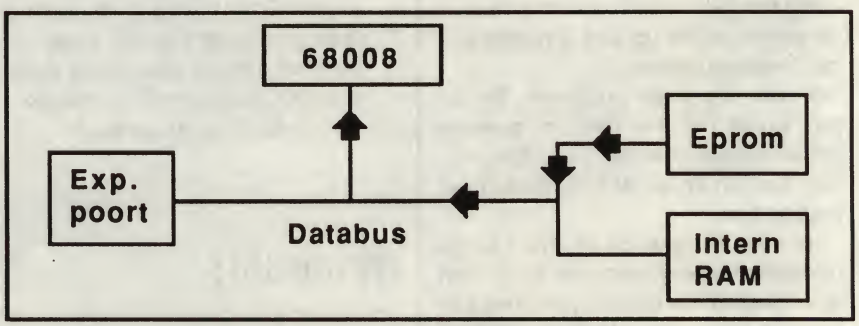
Naar aanleiding van vragen over een bufferprint voor de QL heb ik besloten om hier maar eens een stukje over te schrijven.
Nu heeft Kees van der Wal dit ooit voor mij eens uitgezocht hoe dat nou precies zat. Met die informatie ben ik maar eens aan de slag gegaan.

't Probleem met de datalijnen is dat de informatie 2 kanten op kan gaan op de bus: Bij het schrijven in 't RAM gaat de data van de processor naar RAM en bij het lezen van RAM naar de processor.
Dat zijn al twee situaties die bekeken moet worden.
Bovendien zit een deel van het geheugen in de QL en een ander deel (geheugenuitbreiding / periferie zoals disk-interface) er buiten. Dat levert dus in totaal 4 verschillende situaties op voor de datalijnen van de 68008:

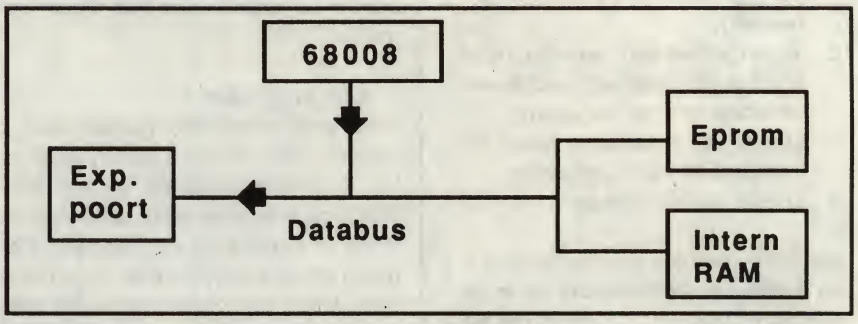
In de gevallen A en B hoeft de bus buffer niets te doen.
In de gevallen C en D moet de buffer natuurlijk wel in aktie komen waarbij de richting van de buffer wordt gestuurd door de processor die via een pootje R/W aangeeft of hij wil lezen of schrijven . Dit 68008 signaal is op de expansie konektor uitgevoerd als signal RDWL (pen 7b).
Een 1 op die poort geeft aan:
lezen door de processor;
een 0 geeft aan:
schrijven door de processor.
Het signaal RDWL kan dan ook direkt worden aangesloten op pen 1 van een 74LS245 (of LS645). Dan moet de A-kant van de busbuffer (pen 2 t/m 9) aan de buitenkant komen en de B-kant (pen 18 t/m 11) met de QL datalijnen verbonden zijn.
Nu moet de buffer nog weten wanneer hij in aktie moet komen. De meest algemene en elegante oplossing is het af te leiden van de "datastrobe" (DSL, pen 6b) en "data strobe master chip" (DSMCL, pen 27a).
Als n.l. intern geheugen wordt aangesproken dan gaat DSMCL tegelijk op en neer met het door de 68008 geleverde signaal DSL.
Externe apparaten moeten, zodra ze aangesproken worden, de lijn DSMCL op +5V houden (Vandaar b.v. de tran-



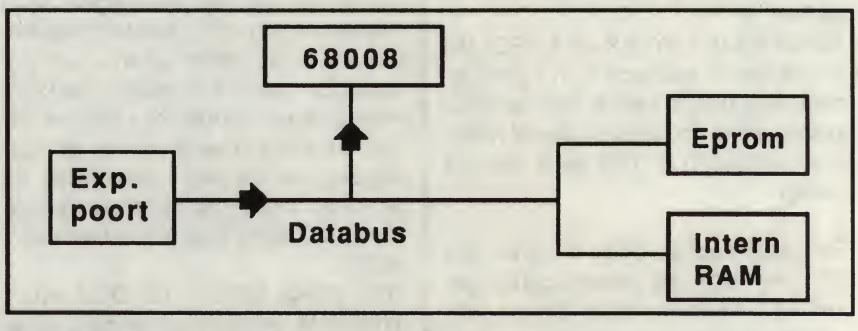
A> Schrijven naar intern geheugen



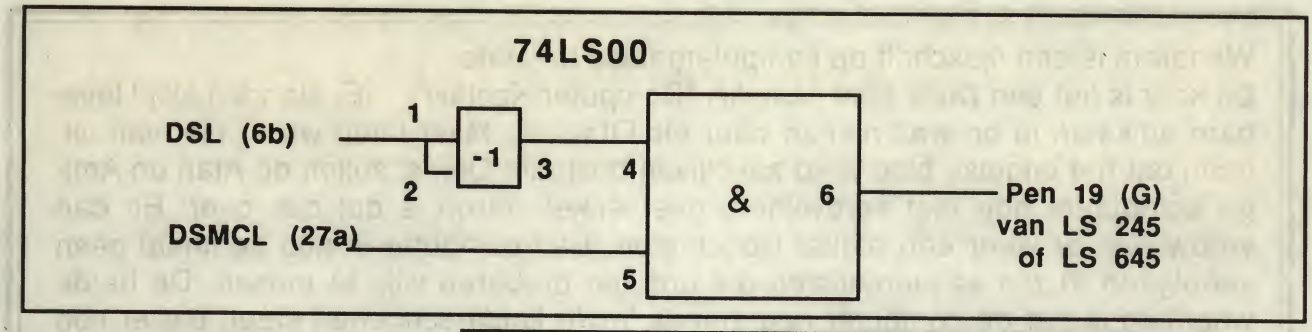
B> Lezen uit intern geheugen



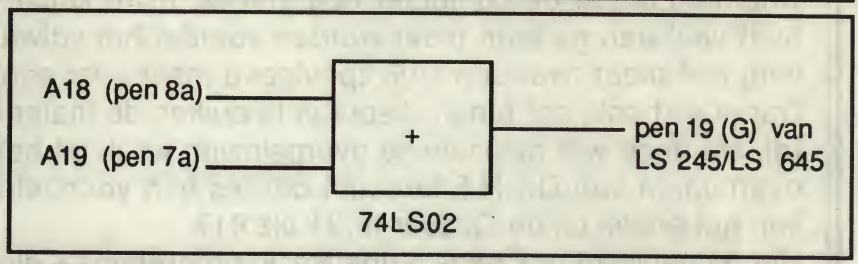
C> Schrijven naar extern geheugen

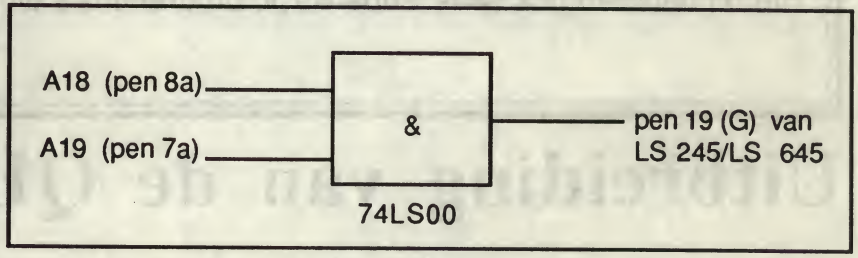


D> Lezen van extern geheugen



sistor op de disk interface e.d.).
Dit schakelingetje werkt goed bij mij en draait al 2 jaar zonder enige problemen. Er zijn ook andere oplossingen die zal ik ook beschrijven maar deze schakelingen heb ik niet uitgeprobeerd of ze feiloos werken, op papier zouden ze moeten werken.
Het aktiveren van de buffer kan ook afgeleid worden van de adresseering. Dan moet er echter onderscheid gemaakt worden in QL's met 128K RAM intern (en waarbij evt. een geheugen uitbreiding aan de buitenkant is aangesloten) en QL's met 512K (of 640K) RAM intern.
Bij een standaard QL met 128K interne RAM of met 512K intern maar met de schakelaar in de 128K stand, zijn alleen de adressen $00000 - $3FFFF intern. De rest is extern en als dus een adres tussen $40000 en $FFFFF voorkomt, dan moet de databuffer in aktie komen.
Dat is eenvoudig vast te stellen aan de hand van adreslijn A18 en A 19.
Bij een QL met intern 640K RAM, hoeft de busbuffer alleen geaktiveerd te worden in 't adresgebied $C0000 - $FFFFF. Bij een QL met 512K intern RAM zou je in principe de adressen $A0000 - $BFFFF nog vrij hebben



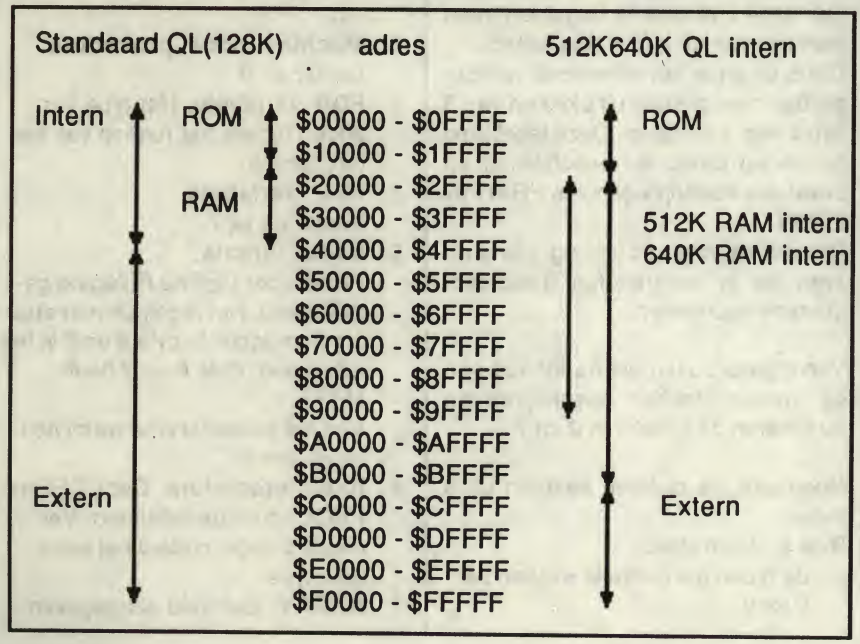


voor 128K externe RAM maar dat is niet waarschijnlijk. (Bovendien zijn de in de handel verkrijgbare 128K uitbreidingen waarschijnlijk op adres $40000 - $5FFFF gesitueerd, zodat die niet gebruikt kunnen worden zonder er het een en ander aan te wijzigen).
Ook dat adres gebied is te detekteren door naar A18 en A19 te kijken.

Als het goed is moet het zo kunnen werken.
Als databus buffer is een 74LS245 geschikt.
Ook de 74LS645 kan gebruikt worden (de pen aansluiting is hetzelfde).
Als men nog meer "power" nodig heeft kan men de 74LS645-1 nemen deze kan 2xzoveel stroom sturen als de 74LS645 of LS245.
De buffering van de adreslijnen is niet zo'n probleem hiervoor kan men gewoon 74LS244 voor gebruiken. Deze gaan maar naar een kant op en kunnen gewoon altijd "aan" staan. Wellicht is het verstandig om (als men een lange kabel (bijv. 1 meter) gebruikt) de adreslijnen met weerstanden af te sluiten. Voor elke adreslijn een weerstand van 680 Ohm naar de +5V aan het eind van de kabel/adresbus.
De weerstanden verminderen eventuele reflekties op de lijn en verminderen ook de overspraak van de ene adreslijn op de andere.

Voor de datalijnen geldt eigelijk hetzelfde, maar dan moet men iets grotere weerstanden gebruiken omdat niet bekend is of alle periferie busbuffers gebruiken. Voor de zekerheid kan men weerstanden kiezen die niet kleiner zijn dan ca. 1K5.
Ik hoop dat dit de zaken wat duidelijker heeft gemaakt en dat men hier verder mee kan solderen.

<Michel>



Wederom is een tijdschrift op computergebied ter ziele.
Dit keer is het een Duits blad namelijk "Computer Kontakt". Er stonden altijd leesbare artikelen in en was niet zo duur als QLworld. Maar laten we er niet van uitgaan dat het engelse blad altijd zal blijven bestaan! Ook al zullen de Atari en Amiga computers nog niet verdwijnen, over enkele jaren is dat ook over! En dan verdwijnen er weer een aantal tijdschriften. Een computer is nou eenmaal geen speelgoed al zijn er journalisten die ons dat proberen wijs te maken. De harde waarheid is dat de computer nog steeds in z'n kinderschoenen staat, dat er nog heel veel aan gedaan moet worden voordat het volwassen is. Daarbij gaat het al lang niet meer over een stuk speelgoed maar over een geavanceerd apparaat.
Dat maakt ook, dat om er dieper in te duiken de materie er niet eenvoudiger opwordt. Na deze wat diepzinnige overpeinzing wil ik uit het laatste CK blad een artikel overnemen van Dr. H.Eltermann dat als een voortzetting beschouwd kan worden van het artikel uit de Quasar nr.31 blz 617.
Hierbij behoren ook twee SuperBasic programma's die hierbij gaan.

Dirk de Vogel

Uitbreiding van de QL set.

Het is U bekend dat men de commandoset van SuperBasic door toevoeging van geschikte machine taal programma's kan uitbreiden.
Men heeft daarvoor een programma nodig, dat met behulp van een assembler in staat is om een procedure zoals PRINT of RUN respectievelijk een functie zoals SIN of SQRT, aan te maken. Dat programma zal ergens in het RESPR-geheugen staan.
Tevens moet men er nog voor zorgen dat men het met een bepaalde naam vanuit SB kan oproepen. En het moet op een bepaalde wijze ingevoerd worden.
De vaders van de QL hebben hiervoor een sub-programma BP.INIT ter beschikking gesteld, wiens oproep adres in deROM op $110 ligt.
De oproep zelf moet in RAM staan, gevolgd door, volgens een bepaald schema vervaardigde definitie lijst.
Die bevat de namen en adressen van het in te voeren machinetaal programma. Een CALL voltooid dan de invoering. Deze methode verdient de voorkeur omdat het gelijktijdig met meerdere programma's mogelijk is.
Het is echter in de praktijk behoorlijk omslachtig en bovendien met enige gebreken behept.
Men kan die nieuwe commando's niet in momentane basic programma's opnemen, behalve misschien in een commando regel.
Dat ligt daaraan, dat het hulpprogramma niet controleerd of die naam reeds aanwezig is, maar het eenvoudig met het nieuwe gebruikers doel aan de naam tabel toevoegd!
De tabel heeft voor SB een speciale betekenis. Die methode laat ook toe dat een naam tweemaal voorkomt, en zelfs voor verschillende doeleinden. Dat functioneert natuurlijk niet.
Eerst met NEW worden alle invoeringen behalve de machine taal codes ongedaan gemaakt. Gelukkig is er nog een andere mogelijkheid die als volgt omschreven kan worden.
Om deze methode te begrijpen moet men iets van de naamtabel weten.
Dat is eigenlijk het adresboek van SuperBasic en bestaat uit blokken van 8 bytes voor elke naam. Deze tabel is na het laden direct ter beschikking en bevat alle inschrijvingen van PRINT tot ERLIN.
Bovendien volgen dan nog alle anderen die in het geladen Basic programma voorkomen.

Vervolgens zullen we de inhoud van de enkele blokken beschrijven en nummeren de bytes van 0 tot 7.

Nogmaals de blokken bestaan uit 8 bytes.
Byte #: Hierin staat:
0: de typen die gebruikt worden van 0 tot 9.
1: type variabele (string 1, vloeiende komma 2, geheel getal 3)
2: relatieve adres van de naam string
3: idem
4: het eigenlijke adresdeel
5: idem
6: idem, de betekenis hangt af van het gebruik.

Type #: Betekenis
---

9 **Machine code functie.** In het adresdeel staat het oproepadres (Het type variabele betekent niets =0)
8: **Machine code procedure.** Verder als 9
7: **FOR variabele.** Het type ver andert tijdens het runnen van het programma
6: **REP variabele.** Verder als bij 7.
5: **Basic functie.** Wordt door DEFine FuNction gedefinieerd. Het regelnummer staat als woordgetal in byte 4 en 5 in het adresdeel. Byte 6 en 7 heeft $FFFF.
Van het variabele type wordt notitie genomen.
4: **Basic procedure.** Door DEFine PROCedure gedefinieerd. Verderals 5 uitgezonderd het variabele type.
3: **ARRAY** Een veld aangegeven

met getallen dimensies. Het type variabele geeft het element aan Het adresdeel duidt relatief op een ARRAY definitie blok in het variabele geheugen.

2: **Variabele** Nader kenmerk van het variabele type. In het adresdeel bevind zich het relatieve adres van het variabele geheugen waarop de waarde van de variabele staat. Is deze variabele nog niet aangebracht dan staat er -1 in.

1: **Getal op de rekenkundige stapel**. Dat wordt slechts intern gebruikt en heeft geen naam.

0: **Een naam**. Hiermee wordt de device naam in het adresboek geplaatst.

Omdat het adresboek in RAM staat, kan het gemanipuleerd worden. Dat moet met verstand gebeuren, omdat SuperBasic zich volledig hierop baseert! Het is derhalve duidelijk wat er gedaan moet worden om nieuwe commando's toe te voegen. Is de gewenste naam bekend dan moet men het overeenkomende blok in het adresboek zien te vinden. Dan kan het te gebruiken type en adresdeel ingevoerd worden, zonder de naamdelen in byte 2 en 3 te veranderen. Het is nu echter wat moeilijk om het daarbij behorende blok in het adresdeel te vinden; omgekeert is het eenvoudiger. Daarom wordt in de volgende programma's een truc gebruikt, die men slechts kan begrijpen, wanneer men weet hoe SuperBasic met deactuele parameters omgaat. Hier handelt SuperBasic na toepassing van deze trucs het gewenste zelf af.

Nu zullen we het programma zelf onder de loep nemen.

Allereerst is er CALL. Het beschikt over twee ingangen, teweten UP en UF voor het invoeren van procedures in machine code respectievelijk functies.
Wanneer men dit programma in het RESPR geheugen heeft, zijn UP en UF tegelijk op de overeenkomende adressen gezet. Voor het invoeren van procedure in machine code met als naam ALFA en op het oproep adres ADR moet men het bevel ALFA=ADR en CALL UP,ALFA invoeren. Daarmee wordt de procedure ALFA gedefinieerd. Voor het invoeren van de machine code functie maakt men gebruik van UF. Hierbij wordt de normale variabele ALFA in een machine procedure ALFA omgezet.

Voor zich is de methode voor het invoeren met CALL volledig toereikend. Om de twee MC procedures DFPR en DFFN in te voeren kan dat in de vorm DFPR NAAM, ADR resp. DFFN NAAM,ADR gebeuren.
Deze beide accepteren alleen namen, die nog geen waarde in het variabele geheugen hebben. Dat lijkt zinvol omdat het verhindert dat een reeds aanwezige variabele omgedefinieerd wordt.

Het beslissende voordeel van beide methoden ligt in de directe beschikbaarheid van het ingevoerde programma.
Het kan bijvoorbeeld voorkomen dat men zich over lange commando namen gaat ergeren. Dan is het mogelijk om bijv. CONTINUE om te zetten met DFPR CONT,32356
Op die manier beschikt men over een commando CONT dat gelijk is aan CONTINUE.
Het zelfde effect krijgt men met CONT=32356 : CALL UP,CONT.
Een verder voordeel bestaat uit de localiseerbaarheid van het ingevoerde programma. Staat er bijvoorbeeld in Basic een met DEFine PROCedure geproduceerde procedure, dan kan daarin met LOC ALFA : DFPR ALFA,ADR een locale machine code procedure toevoegen, die alleen binnen de Basic procedure geldig is!
Na RETurn of END DEFine is ALFA weer als voorheen bruikbaar.
Wie zelf wil uit vissen, hoe de naam tabel in elkaar zit, moet weten dat het begin en het eind adres op de plaatsen 24 (A6) en 28 (A6) in het Basicblok staat en kan met de voortreffelijke BPEEK-commando's van Klaus Gtter (CK 10-11/87) heel gemakkelijk gevonden worden.
Is hiervoor belangstelling dan wil ik dat ook vertalen.

Uit CK 2-3/88. (De laatste)
H.Eltermann
vertaling Dirk de Vogel.

Basicprogramma voor de aanroep van de uitbreiding

```
100 dev$="flp1_"
110 adr=RESPR(512)
120 start=adr
130 LBYTES dev$ & "call_ext",start
140 up=start
150 uf=start+6
160 start=start+40
170 LBYTES dev$ & "dfpr_ext",start
180 dffn=start+6 : CALL up,dffn
190 dfpr=start+6 : CALL up,dfpr
200 PRINT "Variabelen UP en UF vastgelegd!"
210 PRINT "PROCedures DFPR en DFFN"
220 PRINT "ingevoerd!"
```

```
10 REMark ** Basic lader voor machine
**
20 REMark ** code routinen **
30 REMark ** Behorende bij het artikel
40 REMark ** Uitbreiding van de QL set
**
50 REMark ** Gemaakt door H.Elterman
60 REMark ** en Dirk de Vogel **
70 REMark ** CK 2-3/88 (laatste) **
100 dev$="flp1_"
110 som=0 : RESTORE
120 adr=RESPR(512)
130 start=adr
140 FOR i=0 TO 38 STEP 2
150  READ waarde
160  som=som+waarde
170  POKE_W adr,waarde
180  adr=adr+2
190 END FOR i
200 READ check
210 IF som=check THEN
220  SBYTES dev$ & "call_ext",start,38
230 ELSE
240  PRINT "Fout in data"
250  STOP
260 END IF
270  start=adr
280  som=0
290 FOR i=0 TO 66 STEP 2
300  READ waarde
310  som=som+waarde
320  POKE_W adr,waarde
330  adr=adr+2
340 END FOR i
350 READ check
360  IF som=check THEN
370   SBYTES dev$ & "dfpr_ext",start,68
380 ELSE
390  PRINT "Fout in data"
400  STOP
410 END IF
420 DATA 13372,2050,24580,13372,2306,10862
430 DATA 28,20877,12342,-10238,27916,15746
440 DATA -10240,11649,-10236,28672,20085
450 DATA 28916,20085,0
460 DATA 222144
470 :
480 DATA 14908,2306,24580,14908,2050,28672
490 DATA 12342,-18430,27652,28913,20085
500 DATA 8246,-18428,27908,28920,20085
510 DATA 9291,20619,-17461,28650,10827
520 DATA 20621,14456,280,20116,26334,10358
530 DATA -26624,15749,-22528,11660,-22524
540 DATA 28672,20085
550 DATA 373298
```

FUTURA is niet meer

De FUTURA, de SANDY opvolger van de QL is definitief van de baan. In plaats hiervan is men bezig met een uitbreidingskaart voor de IBM zodat het mogelijk wordt QDOS als operation system te gebruiken op de IBM en zijn klonen. Er zijn verschillende versies in voorbereiding, o.a. een met 4 Mb geheugen en een 68020 processor, dit zou de snelheid met faktor 200 verhogen. Hiermee zou ook true multitasking mogelijk moeten zijn.
Zou het dan toch nog leuk worden op een IBM machine???
Toch is de FUTURA als uitbreidingspakket te koop!
Sandy een Italiaans bedrijf met een vestiging in Engeland heeft veel geld geinvesteerd in de ontwikkeling van de FUTURA, en vele onderdelen van de FUTURA waren al gereed, en om toch nog wat geld van de investeringen terug te verdienen worden de ontwikkelde delen als uitbreiding voor de QL verkocht.
Als eerste kwam de Toolkit II van Tony Tebby. Ik neem aan dat deze inmiddels bij een ieder bekend is.
Ook van QRAM zullen de meeste al gehoord hebben. QRAM is overigens ook op de THOR geimplementeerd.
Het Sandy Super-Q-Board voor geheugenuitbreiding, disk- en muis interface is ook al even op de markt.
Blijft over de systeemkast, het toetsenbord en de 4 EPROM uitbreidingskaart.
In de systeemkast is plaats voor de QL, diskinterface, diskdrives en een ruime plaats voor de voeding (geschakelde voeding inbouwen geeft geen probleem) . Reeds ingebouwd is een 5-weg gebufferde uitbreidingspoort.
Het toetsenbord heeft 84 toetsen, 10 functietoetsen en een nummeriek pad.
Ook de mogelijkheid op 4 EPROMS te plaatsen is los te koop.

Meer informatie:
SPEM
Via Aosta 86
10154 Torino
Italia

Prijzen:
Toolkit II (ROM).....£ 29,90
QRAM (ROM).....£ 29,90
Super-Q-Board + muis.....£ 240,00
Systeemkast.....£ 99.90
Toetsenbord.....£ 99.90
QEPROM-board.....£ 34,90

ARCHIVE GEBRUIK

Ik gebruik archive voor het bijhouden van onderdelen in mijn auto, ik ben namelijk service monteur van copieer machines.
Het daarvoor gemaakte menu-gestuurde programma is totaal 39K groot en 40K aan data (komt bij mij overeen met 380 records).

Onderstaand wat informatie en hopelijk wat tips voor nieuwkomers :

- nette opstart

d.m.v. **run object "XXX" ;**
hierin 1 procedure genaamd START ; in deze procedure staan bij mij onder andere de volgende regels
merge object "flp2_XXX" ;
laadt de rest van het hoofdprogramma
sload "flp2_XXX" ;
laadt een eventueel gemaakt scherm in
backup "flp2_XXX_dbf" as "ram1_XXX_dbf";
copieert de data naar ramdisk (moet dus wel aanwezig zijn)
Bovenstaande opdrachten worden ook op het scherm vermeld, met als gevolg dat we niet tegen een leeg scherm aankijken.
De laatste opdracht in START is de aanroep van de procedure strt. Deze opent de file, toont aantal records ect. In strt wordt het hoofdmenu scherm opgestart.
Het voordeel van strt is dat ik na een 'edit' weer opstart door als direct commando strt in te toetsen.
(erg makkelijk bij het maken van een programma) Dus START wordt alleen bij de allereerste opstart gebruikt en elke weder opstart door strt.

Hoofdmenu

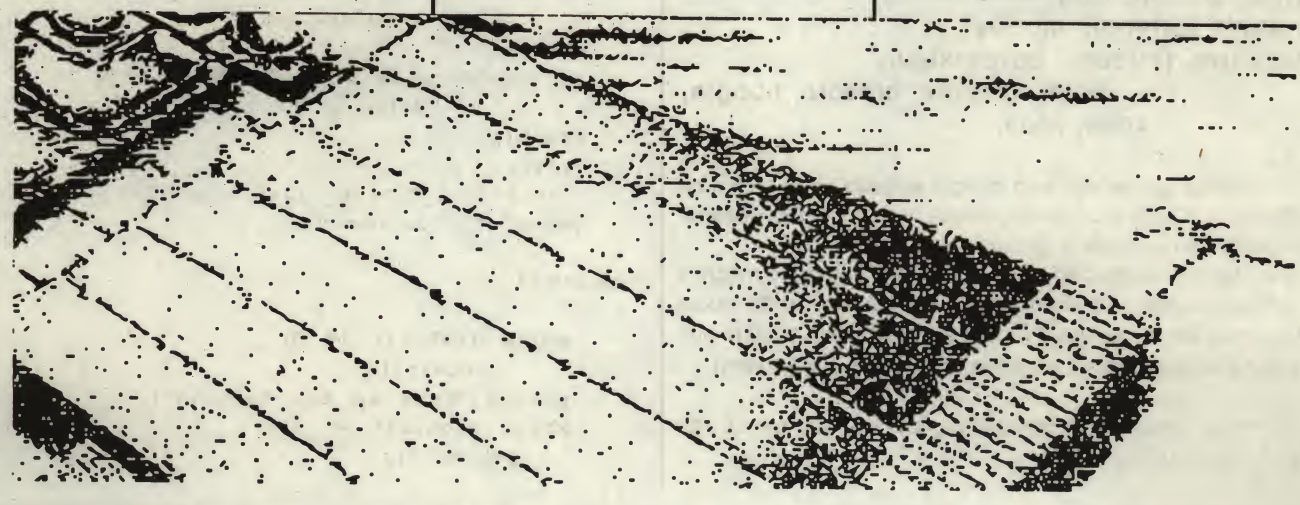
- Het hoofdmenu loopt via een ERROR opvang. Bij een ERROR wordt het error nummer getoond, wordt er gereset (om een event. select op te heffen), opnieuw gesorteerd ect. en weer opnieuw opgestart.
Het programma is dus zelfherstellend. Hierna moet dus eerst de fout hersteld worden, maar het voordeel is dat je niet met een halve of open file zit. Vergeet geen mogelijkheid om het progr. te stoppen.

- Schermgebruik:

Iedereen weet ondertussen wel dat ARCHIVE bijzonder traag is met scherm afhandelingen. Ik gebruik daarom maar 1 scherm met alle recordinformatie op de bovendste helft van het scherm.
De onderste helft is leeg en gebruik ik als window voor vragen e.d. maar ook om file selecties of meerdere records te tonen. Een aantal regels overschrijven gaat sneller dan het omschakelen van schermen.
Op regels 20 t/m 24 (dus onder screen) wordt het hoofdmenu geschreven. Vanuit het hoofdmenu worden andere sub-menu's aangeroepen en deze overschrijft weer regels 20 t/m 24 ect. ect.
- Al de onderdelen hebben een 10 cijferig nummer. Nu zijn er een aantal onderdelen waarbij alleen de laatste 2 nummers anders zijn. Het sorteren gebeurt alleen op de eerste 8 letters/ cijfers. Nu is dit voor mij niet zo'n probleem. Wel is het een probleem dat bij het zoeken naar dit nummer m.b.v. 'locate' (verreweg het snelste zoekcommando), ARCHIVE altijd het eerste nummer met de eerste 8 gelijke cijfers/ letters pakt; het tweede met alleen de laatste 2 anders is met locate domweg niet te vinden. Ik heb dit als volgt opgelost: het programma vergelijkt het gewenste nummer met het gevonden record. Is dit nummer niet exact gelijk dan wordt de dbf nog een keer doorlopen maar nu d.m.v. het commando 'find', nu zal het gewenste record wel gevonden worden. Traag, maar wel effectief . . . Weet iemand hier een betere/snellere oplossing voor. Tot zover mijn ervaringen op ARCHIVE gebruik Vragen of aanmerkingen zijn van harte welkom,

Ed Kats
H.Bosmanslaan 100
3144 LB MAASSLUIS
01899-12594

p.s. Weet iemand een manier om in ARCHIVE/SUPERBASIC te kunnen controleren op floppy-disk naam (dus de naam die een floppy krijgt bij het formatteren), zodat niet per ongeluk een file op een niet gewenst floppy wordt gezet.



QDOS Functies in 'C' voor de Metacomco Lattice C compiler.

Na enige tijd te hebben gewerkt met deze compiler ontstond er de behoefte om vele van de specifieke Qi Qdos functies te kunnen gebruiken zoals Dir, Cls,Exec enz. De compiler is bijna geheel volgens de standaard definitie van Kernighan & Ritchie en de bijgeleverde bibliotheek van functies bevatten zogoed als geen Qi Qdos functies behalve een paar waarmee Qdos traps aangeroepen kunnen worden.

Met behulp van deze functies hebben wij dan ook de hieronder genoemde C functies gemaakt:

exec (filenaam, wait)
fat (stream, line, col)
fborder (stream, breedte, kleur)
fcls (stream, gedeelte)
fcsize (stream, breedte, hoogte)
fcursdown (stream)
fcursleft (stream)
fcursright (stream)
fcursen (stream, modus)
fcursor (stream, xpos, ypos)
fcursup (stream)
fdir (device, bestemming)
fflash (stream, modus)
fink (stream, kleur)
fnextline (stream)
fover (stream, modus)
fpan (stream, aantal_pixels, gedeelte)
fpaper (stream, kleur)
fscroll (stream, aantal_pixels, gedeelte)
fstrip (stream, kleur)
ftab (stream, col)
funder (stream, modus)
fwindow (stream, borderkleur, borderbreedte, breedte, hoogte, xpos, ypos)

De functies geven alle een integer waarde terug welke de standaard Qdos foutmelding bevat, u bent vrij om deze teruggegeven waarde te gebruiken of niet.
Alle functies welke beginnen met een 'f' zijn dmv. macro's ook beschikbaar zonder de 'f' en werken dan op de stdout file [voorbeeld cls()].De funcie exec(filenaam,wait) is ook beschikbaar als macro in de vorm van exec_w(filenaam).

Hier volgt een demoprogramma gelist wat een aantal van de functies gebruikt.

QDOS DEMO C

```
#include <flp1_stdio_h>
#include <flp2_qdos_h>

int _mneed = 5120;
  /* Nodig omdat C programmas al het geheugen
    'opvreten' en nu alleen 5120 bytes. */

main ()                  /* QDOSDEMO_C */
{
    char fnam[36],*fgetchid();
                    /* filnaam array */
    int fout;       /* qdosfout variabele */

    mode(4);
    printf("Dit is een demo van de qdos
            functies\n");
    printf("Stop een programmaschyf in
           drive 1 ");
    pause();
    cls();
    dir("flp1_");
    nextline();
    at(17,0);
    printf("\nGeef de filenaam voor het
            programma om te runnen ");
    scanf("%36s",fnam);
    fout=exec_w(fnam);
    cls();
    window(7,1,512,256,0,0);
    /* Border wit 1 pixel, 512x256 op 0,0 */
    if (fout)
     printf("Qdos fout nummer:%d gevonden ",
                fout);
    else
        printf("Het programma gerund vanuit C
               is afgelopen ");
    pause();
    printf("Stop de orginele disk terug in
              drive 1 ");
    pause();
    cls();
    fdir("flp1_","scr_420x80a46x175");
    printf("Einde demo");
}
pause()
{
    while (kbhit() != 0)
        getchar();
    printf("Druk op een toets\n");
    while (kbhit() == 0);
       getchar();
}
```

Indien uw programma een van deze functies gebruikt moet u aan het begin van uw programma een #include <XXX_qdos_h> waarbij XXX de drive is waar de file staat.
Tevens is bijgevoegd de header file QDOS_H en de linker controlfile QDOSDEMO_LINK.
De procedure die u moet volgen is:
Compileer de QDOS_C file
Rename de output file van de compiler QDOS_o naar de file QDOS_l, dit nu wordt dan de bibliotheekfile van waaruit de functies kunnen worden aangeroepen.
compileer het demoprogramma QDOSDEMO_C
Link nu het geheel met de linker controle file genaamd QDOSDEMO_link.
Alle compilaties en linkings moeten zonder errors en/of warnings kunnen verlopen.
Het demoprogramma doet het volgende:
Kiest mode 4, wist scherm, toont directory van flp1, vraagt om filenaam van een executable file, start dit programma met exec_w, veranderd het window naar 512x256 punten met een witte border van 1 pixel, en toont vervolgens een directory op een deel van det scherm.

Het is geen hoogvlieger van een programma maar het toont wel wat de nieuwe functies kunnen doen.

Veel succes.
W. van Dieren
G. M. van Rooyen

Aangepaste QDOS_H FILE

```
*
*
* The following structure contains the
  registers that are used to interface
* with the operating system via the QDOSx
  functions.
*
*/
struct REGS
   {
    long D0,D1,D2,D3;
    char *A0,*A1,*A2,*A3;
   };
/**
*
* Error codes returned by QDOS
*
*/
#define ERR_NC -1 /* operation not complete
*/
#define ERR_NJ -2 /* not a valid job */
#define ERR_OM -3 /* out of memory */
#define ERR_OR -4 /* our of range */
#define ERR_BO -5 /* buffer overflow */
#define ERR_NO -6 /* channel not open */
#define ERR_NF -7 /* file or dev. not found
*/
#define ERR_EX -8 /* file already exists */
#define ERR_IU -9 /* file or device in use
*/
#define ERR_EF -10 /* end of file */
#define ERR_DF -11 /* drive full */
#define ERR_BN -12 /* bad device name */
#define ERR_TE -13 /* transmission error */
#define ERR_FF -14 /* format failed */
#define ERR_BP -15 /* bad parameter */
#define ERR_FE -16 /* file error */
#define ERR_XP -17 /* error in expression
*/
#define ERR_OV -18 /* arithmetic overflow
*/
#define ERR_NI -19 /* not implemented */
#define ERR_RO -20 /* read only */
#define ERR_BL -21 /* bad line in BASIC */

/*
* the following structure defines the
standard
  QDOS file header
*/

struct FS_HEADR
   {
    long length;       /* file length      */
    char access;       /* file access type
*/
    char type;         /* file type        */
    char info[8];      /* type dependent info
*/
    short name_sz;     /* size of name
*/
    char name[36];     /* name area        */
    char reserved[12];/* reserved for time &
                         date   */
   };
/*
* de volgende zyn macros voor de eigen
uitbreidingen van de lib qdos_l
*/
extern char mstack[]; /* storage room for
mathm. stack */
#define cls()   fcls(stdout,0)
#define dir(a) fdir((a),"scr_")
#define point(a,b) fpoint(stdout,a,b)
#define exec_w(a) exec(a,1)
#define at(a,b) fat(stdout,a,b)
#define border(a,b) fborder(stdout,a,b)
#define csize(a,b) fcsize(stdout,a,b)
#define cursdown() fcursdown(stdout)
#define cursleft() fcursleft(stdout)
#define cursright() fcursright(stdout)
#define cursen(a) fcursen(stdout,a)
#define cursor(a,b) fcursor(stdout,a,b)
#define cursup() fcursup(stdout)
#define flash(a) fflash(stdout,a)
#define ink(a) fink(stdout,a)
#define nextline() fnextline(stdout)
#define over(a) fover(stdout,a)
#define pan(a,b) fpan(stdout,a,b)
#define paper(a) fpaper(stdout,a)
#define scroll(a,b) fscroll(stdout,a,b)
#define strip(a) fstrip(stdout,a)
#define tab(a) ftab(stdout,a)
#define under(a) funder(stdout,a)
#define window(a,b,c,d,e,f) fwindow
(stdout,a,b,c,d,e,f)
```

QDOS DEMO_LINK

```
* Template control file for QLC
*
* Ordering of sections
*
RELOC    startup
SECTION  text
SECTION  data
SECTION  udata
SECTION  end
*
* Start with initialisation code
*
INPUT    FLP1_startup
*
* Now include the user module (from command
  line)
* Any other user modules must be included
  at this point with their own INPUT
  instruction.
*
INPUT    *
*
* Now search library for all undefined
  references
LIBRARY  FLP2_qdos_l
LIBRARY  FLP2_qlc_l
```

Meer weten over C

Indien er belangstelling is om meer over C-uitbreidingen te vertellen zijn wij graag bereid in een volgend artikel hier verder op in te gaan. Aanvullingen op deze uitbreidingen zijn ook van harte welkom, dan kunnen wij gezamelijk een bibliotheek opzetten.
Ook overwegen wij deze uitbreidingen van C onder te brengen bij de Databank of de cartridgeservice, wanneer dat voor U interessant is dan horen wij dat graag .
Ook andere vragen over de programmeertaal C zijn van harte welkom zodat het misschien mogelijk is om een maandelijkse C- Rubriek te starten .
Uw reactie gaarne door middel van een briefje of een telefoontje aan het redactieadres.

Gerard van Rooijen
Wim van Dieren

Redactieadres:

Gruttostraat 15
3435 DJ Nieuwegein
Tel. 03402 - 33027

QDOS C

```
#define MT_CJOB 0x01
#define FS_LOAD 0x48
#define MT_ACTIV 0x0a
#define HEADR 0x47
#define IO_OPEN 0x1
#define IO_FSTRG 0x03
#define IO_CLOSE 0x02

#include <flp1_stdio_h>
#include <flp2_qdos_h>
#include <flp1_fnctl_h>

static char mstack[125];
/* storage room for mathm. stack */

exec(fname,wait)
char *fname;
int wait;
{
     short int timeout= -1;
     struct REGS in,out;
     struct FS_HEADR header;
     int file,fout;
     unsigned long int *dataspace;
     char *chanid,*getchid();

     /* tracht eers de file te openen. */
     file = open(fname,O_RDONLY || O_RAW);
     if ( file == -1 )
          fout = ERR_NF;
     else {

          /* lees nu de header info via
          FS_HEADR */
          chanid = getchid(file);
          in.D0 = HEADR;
        in.D2 = (short int) sizeof(header);
          in.D3 = timeout;
          in.A0 = chanid;
          in.A1 = (char *) &header;
          fout = qdos3(&in,&out);

          /* check de filetype (1 is exec
file) */
          if (header.type != 1)
               fout = ERR_NJ;
          else {
               /* tracht de job te creeren
*/
               dataspace = (unsigned long
*) &header.info[0];
               in.D0 = MT_CJOB;
               in.D1 = -1;
                 /* not independent job */
               in.D2 = header.length;
               in.D3 = *dataspace;
               in.A1 = 0;
               fout = qdos1(&in,&out);
               if (fout == 0 ) {
               /* laad nu de file in
                het geheugen */
```

```
      read(file,out.A0,header.length);

                        /* tracht nu de job te
                           activeren */
                        in.D0 = MT_ACTIV;
                        in.D1 = out.D1;
                        in.D2 = (char) 8;
                        if ( wait )
                             in.D3 = (short) -
1;
                        else in.D3 = (short) 0;
                        fout = qdos1(&in,&out);
                   }
              }
         }
         /* close the file now */
         close(file);
         return(fout);
}

fat(stream,line,col)
int *stream,line,col;

{
   struct REGS in,out;
   in.D0 = 0x10;
   in.D1 = col;
   in.D2 = line;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

fborder(stream,width,colour)
int *stream,width,colour;

{
   struct REGS in,out;
   in.D0 = 0xC;
   in.D1 = colour;
   in.D2 = width;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}



fcls(stream,part)
int *stream,part;

{
   struct REGS in,out;
   switch(part)
   {
      case 0: in.D0 = 0x20;break;
              /* wist gehele window */
      case 1: in.D0 = 0x21;break;
              /* wist boven cursorlijn */
      case 2: in.D0 = 0x22;break;
              /* wist onder cursorlijn */
      case 3: in.D0 = 0x23;break;
              /* wist cursorlijn */
      case 4: in.D0 = 0x24;break;
      /* wist van cursor tot eind cursor-
lijn*/
      default: return(ERR_BP);
      /* foute boel */
   }

   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}



fcsize(stream,width,height)
int *stream,width,height;

{
   struct REGS in,out;
   in.D0 = 0x2D;
   in.D1 = width;
   in.D2 = height;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

fcursdown(stream)
int *stream;

{
   struct REGS in,out;
   in.D0 = 0x16;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

fcursleft(stream)
int *stream;

{
   struct REGS in,out;
   in.D0 = 0x13;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);
   return(qdos3(&in,&out));
}

fcursright(stream)
int *stream;

{
   struct REGS in,out;
   in.D0 = 0x14;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}
```

```
fcursen(stream,modus)
int *stream,modus;

{
   struct REGS in,out;
   in.D0 = 0xE;
   if (modus==0) in.D0 = 0xF;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

fcursor(stream,xpos,ypos)
int *stream,xpos,ypos;

{
   struct REGS in,out;
   in.D0 = 0x17;
   in.D1 = xpos;
   in.D2 = ypos;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

fcursup(stream)
int *stream;

{
   struct REGS in,out;
   in.D0 = 0x15;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}


fdir(device,destination)
char *device,*destination;
{
     int fout,namelen,teller;
     short int timeout = -1,*flen;
     struct REGS in,out;
     struct FS_HEADR header;
     char *chanid,*name,qdosdev
[34],*getchid
     ();
     FILE *outfile,*fopen();

     /* OPEN file naar DEVICE */
     /*************************/

     /* bepaal lengte devicenaam */
     for ( namelen = 0 , name = device ;
           *name != 0 ; name++ , namelen++
) ;
     flen = (short int *)&qdosdev[0];
     *flen = namelen;
     /* maak string qdos compatibel */
     name = device;
     for (teller=2;teller < namelen+2;
          teller++) {
          qdosdev[teller] = *name;
          name++;
     }
     /* laad registers met de juiste waard-
en           voor IO.OPEN
          D1.L JOB ID
          D3.L CODE bit 4=open directory
          A0   ADRES OF CHANNEL NAME */
     in.D0 = IO_OPEN;
     in.D1 = -1;
     in.D3 = 4;
     in.A0 = &qdosdev[0];

     /* roep de trap aan
          TRAP 2 D0=$01 */
     fout = qdos2(&in,&out);
     /* RETOUR IN REGISTERS
          D1 JOB ID
          A0 CHANNEL ID */
     chanid = out.A0;

     /* INDIEN GELUKT DAN */
     /*********************/
     if ( fout == 0 ) {                /* no
error */
          /* OPEN DESTINATION */
          outfile = fopen( destina-
tion,"wa+" );
          /* INDIEN GELUKT DAN */
          if ( outfile == NULL )       /*
oeps er is iets mis */
               fout = ERR_NF;
          else {
               /* LAAD DE DIRECTORY IN HET
                  GEHEUGEN */
               /
************************************/
               out.D1 = (short int) sizeof
(header);
               /* LAAD de REGISTERS
               D2.W lengte v/d buffer
               D3.W timeout
               A0 channel id
               A1 base of buffer */
               in.D0 = IO_FSTRG;
               in.D2 = (short int) sizeof
(header);
               in.D3 = timeout;
               in.A0 = chanid;
               while ( out.D1 == (short
int) sizeof(header)) {
                    in.A1 =(char *)
&header;
                    /* roep TRAP 3 $03
IO.FSSTRG */
                    qdos3(&in,&out);
                    /* COPIEER DIRECTORY
NAAR DESTINATION */
                    /
************************************/
                    if (header.name_sz !=
0)
                         fprintf(out-
```

```
file,"\n%-32s %10ld",&header.name[0],(long
int *)header.length);
                }
          /* close the dest file  */
          fclose(outfile);
          }
     /* close the device file  */
     in.D0 = IO_CLOSE;
     in.A0 = chanid;
     qdos2(&in,&out);
     }
     return(fout);
}
fflash(stream,modus)
int *stream,modus;


{
   struct REGS in,out;
   in.D0 = 0x2A;
   in.D1 = modus;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}
fink(stream,colour)
int *stream,colour;
{
   struct REGS in,out;
   in.D0 = 0x29;
   in.D1 = colour;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);
   return(qdos3(&in,&out));
}
fnextline(stream)
int *stream;

{
   struct REGS in,out;
   in.D0 = 0x12;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}
fover(stream,modus)
int *stream,modus;

{
   struct REGS in,out;
   in.D0 = 0x2C;
   in.D1 = modus;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}
fpan(stream,distance,part)
int *stream,distance,part;

{
   struct REGS in,out;
  switch(part)
   {
      case 0: in.D0 = 0x1B;break;
       /* verschuift gehele window */
      case 3: in.D0 = 0x1E;break;
       /* verschuift cursorlijn */
      case 4: in.D0 = 0x1F;break;
       /* verschuift rechts van cursorlijn
*/
   }

   in.D1 = distance;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

fpaper(stream,colour)
int *stream,colour;

{
   struct REGS in,out;
   in.D0 = 0x27;
   in.D1 = colour;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

fscroll(stream,distance,part)
int *stream,distance,part;

{
   struct REGS in,out;
   switch(part)
   {
      case 0: in.D0 = 0x18;break;
         /* scrollt gehele window */
      case 1: in.D0 = 0x19;break;
         /* scrollt boven cursorlijn */
      case 2: in.D0 = 0x1A;break;
         /* scrollt onder cursorlijn */

   }
   in.D1 = distance;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}
fstrip(stream,colour)
int *stream,colour;

{
   struct REGS in,out;
   in.D0 = 0x28;
   in.D1 = colour;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}
```

```
ftab(stream,col)
int *stream,col;

{
   struct REGS in,out;
   in.D0 = 0x11;
   in.D1 = col;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}

funder(stream,modus)
int *stream,modus;

{
   struct REGS in,out;
   in.D0 = 0x2B;
   in.D1 = modus;
   in.D3 = -1;
   in.A0 = (char*)fgetchid(stream);

   return(qdos3(&in,&out));
}


fwindow(stream, bordercolour, border-
width,                  width, height,
xpos, ypos)
int *stream, bordercolour, borderwidth,
       width, height, xpos, ypos;
{
 short int xy[4];
  struct REGS in,out;
  xy[0] = width;
  xy[1] = height;
  xy[2] = xpos;
  xy[3] = ypos;
  in.D0 = 0xD;
  in.D1 = (char) bordercolour;
  in.D2 = (short) borderwidth;
  in.D3 = (short)-1;
  in.A0 = (char *)fgetchid(stream);
  in.A1 = (char *)&xy[0];

  return(qdos3(&in,&out));
}


mode(x)              /* SETS SCREENMODE 0
of 8 */
short int x;

{
     struct REGS in,out;
     in.D0 = 0x10;
     in.D3 = (short) -1;
     in.D1 = (short) x;
     in.D2 = (short) -1;
 /* READ DISPLAY TYPE */
     return(qdos1(&in,&out));
}
```

Terugkijkend lijkt ARCHIVE plots een beetje zielig:

FLASHBACK

Van de vier XCHANGE-programma's die iedere QL-gebruiker bij zijn toestel cadeau kreeg, spreekt ARCHIVE bij sommigen het meest tot de verbeelding. Het is en blijft dan ook een zeer krachtig systeem, met zijn multifile-capability, zijn screen layout editor, en vooral zijn gestructureerde macrotaal.

Evenwel, naast de diverse bugettes die zelfs in Archive 2.36 nog steeds aanwezig zijn, en het ontbreken van een serieus mailmerge-systeem (de halfslachtige programmatuur die sommige third-party softwarehuizen versjacheren niet te na gesproken) heeft het hele systeem een grote beperking: zijn T-R-A-A-G-heid. Een deel van de blaam kan op de macrotaal gestoken worden, die uiteraard als interpreter draait (dit is overigens ook het geval bij dBase III en III+). Dit speelt vooral mee voor wie allerlei dynamische screen-formatting foefjes wil uithalen, maar niet voor de database-operaties stricto sensu (zoeken, extraheren, sorteren,..). Hiervoor is de grote beperking, dat het hele klereding rechtstreeks op de file werkt, als sort-merge database. Een voordeel hiervan is, dat Archive weinig geheugen verbruikt (maar wacht tot je meerdere orderings en selects op een grote file opeenstapelt): de snelheid is er dan ook wel naar. Het leed kan wel worden verzacht door de hele _dbf-file naar ramdisk te transfereren voor gebruik, en naar floppy na updaten, maar echt snel is nog steeds anders (je moet immers nog steeds via het QDOS filing system passeren).

Multitasking is en blijft een nationale ramp, tenzij grof geschut als Taskmaster, QRAM, of JAM (Job Applications Manager) wordt ingeschoten. Ook zit men vastgenageld aan fixed format records: zo is het bv. een heksentoer om met een enigzins aanvaardbare databank voor boeken en tijdschriftartikels door elkaar voor de dag te komen. En de screen- routines: 't is eraan te zien dat XCHANGE op terminals van een mainframe werd ontwik-

keld.
Een slak met AIDS lijkt soms nog razendsnel !!!
SPEEDSCREEN biedt soms de oplossing, maar veroorzaakt dan weer andere problemen (bij mij: crash boem frut !!).

AANPAK

Sector Software, makers van solide produkten als Taskmaster en Spellbound, besloten dat hier nog een gat in de markt lag. Gebruikmakend van de onvolprezen TURBO-compiler en de bijbehorende dito Toolkit, schreven zij een nieuw database-pakket, genaamd FLASHBACK. Het komt uit voor de in feite belachelijke prijs van £25 (vergelijk even met sommige flutgames !), en multitaskt probleemloos.

De filosofie van het produkt staat haaks op die van Archive .
Snelheid, flexibiliteit van het bestand en bedieningsgemak primeren.
Overwegende dat elke zichzelf respecterende QL-gebruiker nu wel een geheugenuitbreiding zal hebben (bv. de miraculeuze Trump Card), werd daarom voor een memory-resident aanpak gekozen. Zodat ook voor search,sort en merge, SEARCH_MEMORY, MOVE_MEMORY resp. de snelle stringbehandeling van Turbo hun beste beentje konden voorzetten.

HAAS EN SCHILDPAD

Wegens de memory-residente opbouw, stond ook niet in de Grondwet van 1776, dat de records een vast formaat moesten hebben. De auteur (Peter J.Jefferies, van o.m. Taskmaster en Spellbound) implementeerde dan ook een variabele recordstructuur: je kan in elk record fields bijvoegen of wegdoen naar believen, en zelfs een sub-recordstructuur definieren (bv. voor adressen 'op het werk': departement, gebouw, binnenlijn,...).
De makers claimen dat de hele affaire 50 a 100 keer sneller gaat dan Archive. Wel, ik heb de zaak een keer getest op een file met 450 records (75 K), en zou het nog geloven ook. Searches zijn gedaan voor je goed en wel de entertoets hebt losgelaten ! Sorts gebeuren terwijl je even een oogje pinkt naar de buurvrouw (vroeger kon ik op diezelfde file probleemloos sorteren terwijl ik de koffie opzette, of omgekeerd al naargelang het standpunt (dan nog op ramdisk: op disk had ik nog probleemloos kunnen wachten tot hij klaar was en hem uitdrinken !). Groupings (selecties) gebeuren even waanzinnig snel.

HELLO GOODBYE

Ook multitaskt de hele affaire met de ogen dicht. Bij het opstarten wordt het hele ding resident gemaakt (default windowsizes en databases kunnen worden ingepatched met CONFIG_BAS). Even ALT \ drukken (een andere toets kan worden geprogrammeerd) en het ding komt op, met real windows (geen QRAM meer nodig !). ESC en het programma gaat even een dutje doen, terwijl u voortwerkt aan wat u bezig was (bv. een paper schrijven). Heel plezant is het TRANSFER-commando: met CTRL-T kan het huidige record (of elk field daaruit) via de keyboard queue worden 'ingetypt' in het lopende programma (nooit meer zitten mieren met kleine stukjes papier als je een adres of referentie moet opzoeken !). Snel is het systeem nu wel niet, en QRAM krijgt uiteraard waarschijnlijk meteen een indigestie.
Maar voor het gestelde doel voldoet het uitstekend.

Voor een ding verdient de programmeur wel een pak billenkoek: het gebruik van CTRL-C als commando om een record te creeren. Wie nog een beetje wil multitasken, moet dan in zijn boot een ander karakter definieren als window-omschakelaar door POKE_W 163986,c, waarbij c de code voorstelt van het karakter (bv. 31 voor CTRL-SHIFT-ESC>.

Om Archive dbf's te converteren, met of zonder wijziging van de recordstructuur, is een programmaatje bijgeleverd waarmee een kind de was kan doen. Dat zou dus geen beletsl mogen zijn. Het boeleke lijdt klaarblijkelijk wel nog aan kinderziekten: een _dbf van 70 k werd wel in een dikke minuut geconverteerd, maar bij een tweede poging (met 30 nieuwe records bijgevoegd) slikte hij wel de nieuwe records in, wat ik ook probeerde. Tot ik ontdekte... dat ik vergeten was opnieuw te exporteren. Zo moeilijk was het nu ook niet geweest om rechtstreeks van de file af te werken.

COMING SOON

Globaal vertoont het hele systeem een groot gebrek: de beperkte besturingsmogelijkheden. OK, supersnel interactief werken is goed en wel, maar wat als je wil mailmergen, mooie geformatteerde listings produceren,...? Wat dat betreft, kan je best Archive nog even houden. De auteur wist dat ook wel, en kondigt daarom twee nieuwe produkten aan: eerstens een Superbasic-interface (hetgeen alles zou oplossen, en waarschijnlijk ook zal toelaten de eigen databank-applicaties met TURBO te compileren. Zulk een systeem zou vrijwel uniek in de micro en PC wereld zijn qua performantie en veelzijdigheid, en in feite de uitwerking zijn van Stephan Verbeecks concept uit een ouder kilobytje. Tweedens (en logisch aansluitend hierop): een report generator systeem, dat alle mogelijke mailmergeachtige toestanden de baas moet kunnen en nog veel meer.
Het genereren van bv. een verslag, formulier,... in het juiste formaat vanuit Archive gaat wel, maar vereist grote handigheid met de commandotaal, terwijl van flexibiliteit (snel op zicht iets wijzigen aan de layout) hoegenaamd geen sprake kan zijn. Juist reportgeneratie is nu het grote (en mijn bescheiden inziens het enige significante) voordeel van dBase III over Archive. Kan dit opgevangen worden, heeft de professionele QL-gebruiker (en die zijn er!) weer een reden minder om naar allerlei nagemaakte IBM-schroot over te schakelen.

Wij wachten vol interesse op beide produkten. Ondertussen zal voor minder arcane databank-toepassingen Flashback nu reeds gouden diensten bewijzen aan o.m. ondergetekende.

Uit Kilobytje

CURSUS MACHINETAAL Hoofdstuk 8

SHIFTS

Een aspect van Machine Code Programming hebben we nog niet behandeld; het schuiven van data. Vooral in rekenkundige berwerkingen is schuiven belangrijk.

Zoals we al hebben gezien bij binair rekenen, kunnen we de getallen 2 en 4 respectievelijk voorstellen door %00000010 en %00000100. Het bit schuift n positie naar links. Vermenigvuldigen we 4 opnieuw met 2, dan zien we dat %00001000=8 ontstaat. Hela, dat zou betekenen dat het naar links schuiven van een byte over 1 positie overeenkomt met het vermenigvuldigen met twee!. Werkt dit ook voor andere getallen?
%00010011 shift left geeft
%00100110
19 x 2= 38
Ja dus. U ziet al dat voor het naar links schuiven in het engels Shift Left wordt gebruikt. Het blijkt dat we kunnen vermenigvuldigen met 2^n door een bepaald byte n posities naar links te schuiven. 7x8 is dan dus 7 binair geschreven, en dan 3 posities (8=2^3) naar links geschoven.
Aan de rechter zijde worden er nullen 'ingeschoven'.
Het resultaat is dan
%00111000=56.
Dit werkt natuurlijk alleen als er rechts nullen worden ingeschoven. Hiervoor is een speciale instructie: ASL; arithmetic shift left.

Kunnen we ook naar rechts schuiven? Ja; dan vermenigvuldigen we niet met twee, maar delen we door twee.
De deling die wordt uitgevoerd hoeft natuurlijk niet te 'passen' want 7/2=3 rest 1. Voeren we een shift right uit op het getal %00000111, dan wordt de meest rechtse '1' eruit geschoven. In het conditie code register, ook wel status register genoemd (zie beschrijving van de processor) is een bit gereserveerd. Dit bit heet carry bit. Het bit dat bij een shift operatie uit een operand 'valt' wordt 'opgevangen' in het carry bit.

Zo zal een shift left van
%10001000 het resultaat
%00010000 opleveren, met een carry op '1' gezet. Met behulp van branch instructies kunnen we dan springen als we dit wensen:
Branch if Carry Set(BCS).
Ook kunnen we springen als het bit niet gezet is; dan gebruiken we de instructie Branch if Carry Clear (BCC). Deze laatste opdracht zal uitgevoerd worden als we bijvoorbeeld %01101101 naar links schuiven. Dan komt er een '0' in het carry bit van het SR te staan. Clear is het engelse woord voor 'schoon'. 'Niet gezet' betekent 'niet op "1"' en dus schoon, clear.
Zouden we %01101101 echter naar rechts schuiven, dan wordt het carry bit op '1' gezet, en wordt de opdracht BCC niet uitgevoerd. (carry is niet 'clear')
Er zijn verschillende methoden om te schuiven. We kunnen een shift operatie uitvoeren, waarbij we altijd een '0' inschuiven. Dit zijn de logical shift operaties: LSR en LSL voor rechts- en links schuiven respectievelijk. We kunnen ook het bit dat er rechts (of links) 'uitvalt' er respectievelijk links (of rechts) weer inschuiven. We doen dan net of het linker deel van de data vast zit aan het rechter deel, en 'roteren' dan de inhoud. Dit zijn de Rotate operaties: ROR en ROL voor Rotate Over Right (roteer rechts) en Rotate Over Left.
Voorbeeld:
Rotate Over Right van
%11000110 levert %01100011 op.
De '0' die er rechts uit is geschoven wordt er links weer in geschoven. Tegelijkertijd wordt er echter ook een kopie van het bit gemaakt dat deze reis maakt: naar het carry bit. In dit voorbeeld zal de '0' dus naar het carrybit gekopierd worden. We kunnen dan weer een actie ondernemen met de instructies BCC of BCS als we dit wensen.
Zojuist heb ik de ASL en ASR operatie genoemd. Hier is iets speciaals mee aan de hand. De ASL operatie schuift, net als de LSL, een '0' rechts in de data die we onder handen hebben. De ASR echter, kopiert het meest linkse bit en schuift dat opnieuw links naar binnen. Een ASR van %00010010 levert dus (de '0', het meest linkse bit wordt gecopierd) %00001001 op. Rechts valt er een '1' uit, en die gaat naar het Carry bit in het SR. Een ASR van %10001010 levert %11000101 op!. Het meest linkse bit, een '1' wordt gekopierd, en opnieuw links ingeschoven. De '0' die er rechts uit valt, gaat weer naar het C bit van het SR.
Het kopieren van het meest linkse bit heeft tot gevolg dat het teken van de data bewaard blijft. Ik heb U nog niet uigelegd hoe negatieve getallen worden weergegeven. Als we gebruik maken van het zogenaamde twee-complements stelsel, is een negatief getal te herkennen aan zijn eerste bit. Als een getal uit 8 bits bestaat, gebruiken we 7 bits om een getal aan te geven, en 1 bit voor het teken. Het meest linkse bit is het teken bit. Is dit bit '1' dan gaat het om een negatief getal. W hebben nog 7 bits over, en kunnen dus nog de getallen 0 to 127 maken. Met het tekenbit kunnen we '+' of '-' aangeven.
Totaal kunnen we nu dus met 8 bits van -127 tot +127 tellen. 127 wordt voorgesteld als %01111111, aftellend komen we tot %00000001 voor het getal 1, %00000000 betekent het getal nul, en als we verder gaan aftellen, is het getal -1 gelijk aan 0-1. Aftrekken gaat door te 'lenen' bij het 9e bit. Dit zit niet in het byte, maar denken we erbij.
De aftreksom wordt dan:

```
bit 98   0
     %100000000
     % 00000001-
     ----------
     % 11111111
```

Min 1 wordt dus door 8 '1'(nen) voorgesteld. We kunnen verder naar beneden aftellen. -2 is -1-1=

```
%11111111
%00000001-
----------
%11111110
```

We zien dat het eerste bit '1' blijft, en dus een '-' teken aan geeft. Zo kunnen we verder aftellen tot -127. Dit wordt voorgesteld als %10000001. Probeert U eens te rekenen met wat negatieve en positieve getallen door elkaar. U zult zien dat U geen moeite heeft met aftrekken en optellen. Het aftrekken is gewoon het optellen met een negatief getal.
Het omrekenen van een positief naar een negatief getal gaat eenvoudig: U neemt de binaire voorstelling van het getal dat U negatief wilt maken. Van iedere 1 maakt u een 0 en omgekeerd. Dit heet inverteren. Vervolgens trekt U 1 van het verkregen getal af. Voorbeeld: -3 is het getal 3 binair geschreven: %00000011 dit getal inverteren we: maak van 1->0 en 0->1: %11111100 vervolgens tellen we 1 bij dit getal op:

```
%00000001+
----------
%11111101
```

Het antwoord kunnen we controleren door de optelling -3+3 uit te voeren. Hier moet 0 uitkomen:

```
%00000011
%11111101
----------
%100000000
```

We zien dat dit klopt, maar dat we een 'carry' hebben. Dit is echter precies het 9e bit dat we eerder hadden bedacht om van te kunnen lenen toen we van 0 het getal '1' wilden aftrekken. Het op deze wijze rekenen met negatieve getallen heet, zoals eerder genoemd het two's complement systeem.
We kunnen bij de shift operaties niet alleen bit-voor-bit schuiven, maar ook met meerdere bits tegelijk.
Als we het getal dat in D0 staat vijf plaatsen naar links willen schuiven, dan gaat dat zo: LSL #5,D0
Dit (immediate mode) kan tot maximaal 8 bits. Willen we meer bits schuiven, dan kunnen we het gewenste aantal in een ander dataregister zetten, en vervolgens de opdracht geven te schuiven. Het heeft natuurlijk weinig zin om meer dan 31 bits te schuiven, omdat bij bijvoorbeeld een LSL het gehele dataregister met '0' gevuld is. D0 12 plaatsen naar links schuiven gaat als volgt:

```
MOVE.B    #12,D1
LSL.L     D1,D0
```

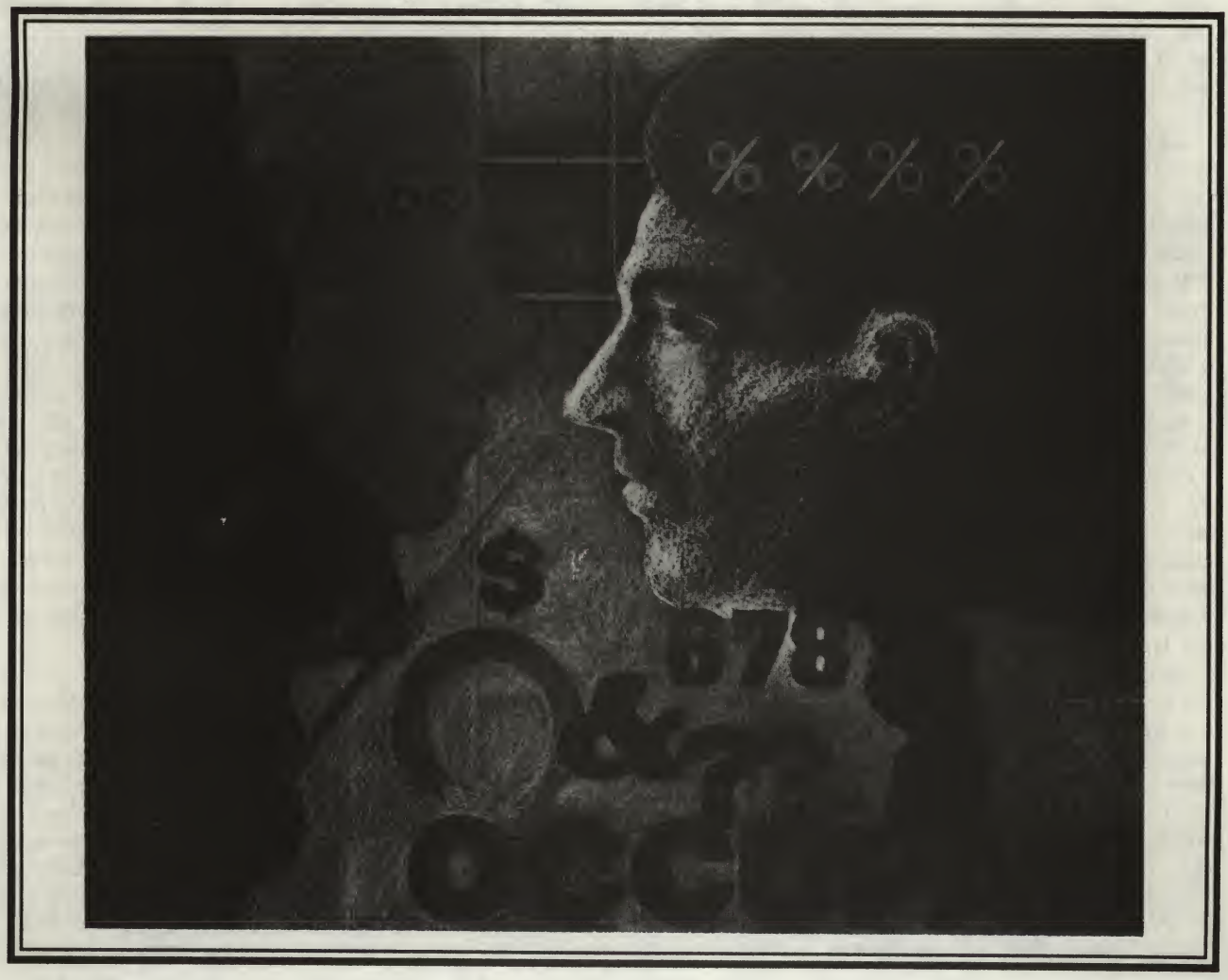
We plaatsen het getal 12 in register D1, daarna moet de inhoud van register D0 met de inhoud van D1 (zijnde 12) naar links worden geschoven. Alle 32 bits schuiven, omdat we de toevoeging '.L' hebben gegeven.
Het carry bit kan natuurlijk maar 1 bit bevatten. Daarom wordt bij een shift van meer dan 1 bit alleen het bit bewaard dat het laatst uit de data wordt geschoven.

```
LSL.L     #3,#%01110111
```

(geen toegestane instructie overigens, dit zou ook weinig zin hebben!) wordt dus %00001110 met een '1' in het carry bit.

```
LSL.L     #4,#%01110111
```

geeft %00000111, met een '0' in het carry bit.



PRO FORTRAN 77:

FORTRAN (FORmula TRANslator) werd reeds in 1954 door de Amerikaan J.Backus ontwikkeld, en is hiermee de eerste hogere programmeertaal die algemene acceptantie mocht kennen. Zoals de naam zelf zegt, werd zij vooral ontworpen met het oog op de toepassingen, waarvoor uiteindelijk ook de eerste computers werden geconstrueerd: zuiver numeriek rekenwerk. Faciliteiten voor bestandsbeheer waren bijzaak.
Ook stonden zgn.parsing algoritmes nog in veel minder dan kinderschoenen (ALGOL, waaruit later PASCAL ontstond, was de eerste taal waarvoor parsing en formele syntax werkelijk ernstig gebruikt werden), zodat een vrij rigiede, beknopte syntax gebezigd werd. Zo worden alleen de kolommen 7 t.e.m. 72 toegestaan voor statements, moeten voortgezette lijnen worden aangeduid met een teken in kolom 6, en kan men slechts een statement per regel zetten.

Deze eigenschappen, naast het sterk lineaire karakter van de gebruikelijke (en meest efficiente) Fortranschriftuur zijn de grond voor smalende bijnamen als 'de GOTO-taal' of 'dat stuk macro-assembler' uit de mond van gestruktureerde adepten, vooral van hen die bij PASCAL zweren. Veel van deze kritiek berust op zure druiven, maar zoals Henry Ford reeds zei:" geen idee is zo dwaas of er zit wel iets van waarheid in". Het is inderdaad zo, dat FORTRAN in vele opzichten een uitgesproken 'low-level language' (i.e.: dicht bij de machine staande taal) is, zo sterk, dat er bijna de een-tot-een correspondentie tussen source en object code bestaat die typerend is voor assembly language. Dit echter is voor de geschetste toepassing eerder een kracht dan een zwakte: compilers kunnen zodanig efficiente code genereren dat zij weinig trager loopt dan een origineel machinetaalprogramma. Nadeel is dan weer, dat 'gestruktureerde' concepten als WHILE-ENDWHILE,DO-BEGIN-END,... geheel ontbreken: programma's van door de wol geverfde Fortranisten lijken welhaast op een spaghetti van labels, GOTO's, computed GOTOs, IF-GOTO's etc...

Op andere gebieden wordt men juist weer verwend. Mathematische mogelijkheden zijn groter dan in welke taal ook, op basis van 'schrijf-zoals-je-spreekt'. En geen andere taal heeft zodanig transparante voorzieningen voor geformatteerde output. Zo volstaat om een M op N tabel in mooie kolommetjes (bv. 2 plaatsen na de komma) te laten afdrukken eenvoudigweg:

```
DO 10 I=1,M
10    WRITE(*,'(N(2X,F8.3)')(A
(I,J),J=1,N)
```

Terwijl in PASCAL alleen al om de afdruk op 2 plaatsen na de komma van een getal te doen je 3 of 4 procedures moet ineenklooien.

Voor geen enkele taal bestaan bovendien zodanig effectieve optimalisatie-algoritmes (compilers met 3 of 4 optimalisatieniveaus zijn op mainframes gemeengoed).

Serieuze gebreken waren er ook wel. Zo bv. waren stringvariabelen niet voorzien (maar door een hoedentruc kan je wel vier letters in een integer inlezen en verwerken; net zoals later C laat Fortran bijna alles toe wat naam heeft aangaande type-coercions).
IF-THEN-ELSE bestond niet.

Maar in de nieuwste ANSI-standaard (American National Standards Institute, zoiets als ISO in Europa), FORTRAN 77, zijn deze beperkingen goeddeels opgeheven. Het type CHARACTER werd ingevoerd, samen met bewerkingen ermee die in PASCAL al een halve library vereisen. Voor de rekenfreaks werd het type COMPLEX aan de specificatie toegevoegd.
Bit-manipulaties (SHIFT,AND,OR,...) zijn tegenwoordig gemeengoed. Alles samen maakt, dat FORTRAN als hoogperformante taal alleen van C ooit concurrentie te vrezen zal krijgen (en dan zal men nog een wreed goeie library voor de derde letter moeten schrijven).

Op micro's wordt deze taal een beetje ondergewaardeerd: echte rekenaars werken meestal op mini's of mainframes, terwijl informaticastudenten om didactische redenen de voor iets anders eigenlijk te beperkte Pascal wordt opgedrongen. Niettemin is er wel een zekere vraag naar Fortrancompilers voor micro's, voor mensen die beroepshalve veel met de taal te maken hebben of gaan krijgen. Op vraag van enkele van deze mensen ging de club op zoek.

En vond... Pro Fortran 77 van Prospero, voor de zachte (ahem) prijs van ca. 7000 BFr. Ondergetekende, beroepshalve Fortranist tot in de kist, offerde zich op als proefkonijn.

De package bestaat uit een manual van ca. 200 kleingedrukte A5-pagina's, waarin niet alleen de taal volledig wordt beschreven, maar ook veel nuttige informatie over de implementatie gegeven. Zelfs een formele syntax-beschrijving (voor Fortran eigenlijk overbodig) werd opgenomen. Het werk is wat droog-pedant, maar niettemin duidelijk geschreven.

De compiler zelf komt op drie cartridges, van iets betere kwaliteit dan we gewend zijn. Een EPROM bevat het runtimesysteem, en is vereist voor compilatie zowel als uitvoering (voor het laatste kan ook een runtimesysteem bijgelinkt worden). Het programma is niet beveiligd en - als je de manual leest - vrij eenvoudig te herconfigureren voor disks.

De compilatie gebeurt (zoals gewoonlijk op micro's) in twee fasen: de parsing (of syntactische analyse) en de codegeneratie. Een aantal opties zijn beschikbaar, zoals runtime checking op bereiken van arraydimensies en functie-argumenten, bijhouden van lijnnummers en identifiers voor runtime foutmeldingen, speed of space optimalisatie, en listing-opties (onge-

+ hexadecimale object-map, variable map (=cross-reference),...) Ook kan de default-grootte voor integers op 16 of 32 bits worden gesteld.
Foutmeldingen zijn duidelijk, en aangeduid tot op het karakter nauwkeurig. Een editor is er NIET: gebruik Quill (aarghh!), The Editor, Sedit (het mini-editortje van Tony Tebby uit de Quanta-library: voldoet uitstekend voor Fortran !),Metacomco ED,....

Overleeft uw programma de compiler, dan volgt de alomtegenwoordige GST-linker, die bij uw programma de vereiste library-routines bijlinkt (in dit geval veelal de mathematische functies), en u eventueel toelaat uit verschillende source files een object file te maken. Pro Pascal en Pro Fortran programma's kunnen naar believen worden gelinkt. In tegenstelling tot de watergruwel van Metacomco C, duurt het linken hier slechts ca. een kwartminuut.

Vervolgens kan het programma worden uitgevoerd. Drie prompts worden aangeboden: default input, default output en optiestring. Wie TOOLKIT II gebruikt, kan ze alledrie met het EX of EW statement meegeven. Default devices en het al of niet aanbieden van de prompts kunnen worden gepatched.
Qua uitvoeringssnelheid klopt de code alle records. Zij is sneller dan die door Turbo gegenereerd (nochtans zelf al een Porsche onder de QL-talen), en dubbel zo snel als Metacomco C. De screen handler gaat zo snel, dat de QL het amper kan bijhouden. De 32-bits floating point berekeningen (weinig minder nauwkeurig dan de inwendige precisie in superbasic) laten de QL eindelijk in zijn volle numerieke glorie zien, terwijl 64-bits aritmetiek volstaat voor zelfs de meest pedante krententeller.

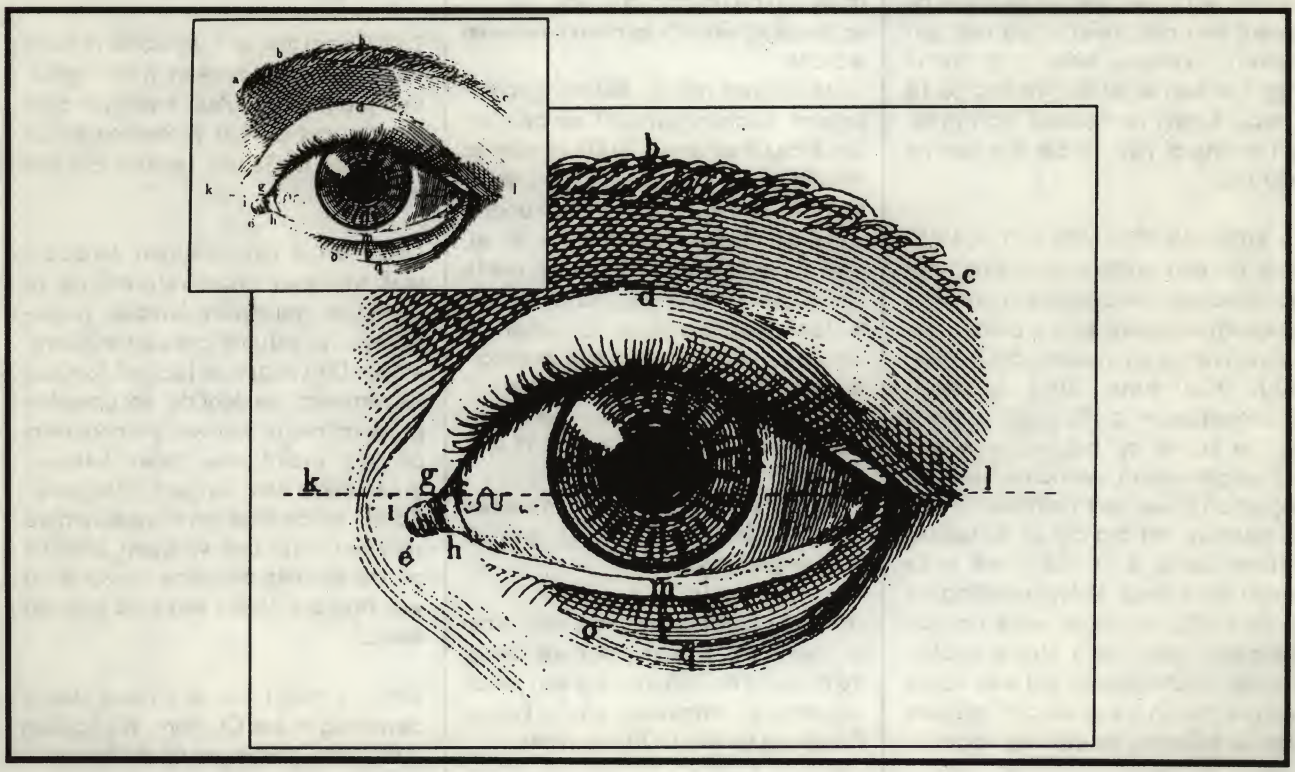
Wat de implementatie zelf betreft, wordt de volledige ANSI standaard gesupporteerd. Bovendien wordt een comprehensieve set QDOS-functies meegeleverd, zodat ook het scherm ten volle kan worden gebruikt (in Metacomco C moet je dan al een hele library beginnen samenstellen, om van het niet beschikbaar zijn van cursortoetsen in de standaard input aldaar nog maar te zwijgen). Ook de meest verspreide niet-standaard library-functies zoekt u niet vergeefs. Varabelen van vrijwel elk type kunnen vrijwel elke vereiste woordlengte krijgen. IPEEK en POKE laten gefoezel toe, GETCOM geeft toegang tot de command line, EXECPG laat toe vanuit het programma een ander te starten, INCLUDE geeft eindelijk de veelgeprezen faciliteit van C om 'header files' met diverse declaraties apart te verwerken, en waar nodig in het programma te specifieren zonder hertypen (deze faciliteit bestaat ook in VAX Fortran 77 op de DIGITAL-mini's en mainframes). In een dergelijk filetje kunt u bv. definities van numerieke constanten, PARAMETERs van arraygrenzen, en veelgebruikte COMMON en EQUIVALENCE declaraties stoppen.

Een library-manager is bijgeleverd, zodat routines uit programma's getrokken kunnen worden als kiezen.

Compilaties verlopen vrij snel; programma's van 10.000 lijnen en meer worden zonder morren verwerkt. Bugs heb ik tot hiertoe niet kunnen vinden. Numerieke overflows at runtime worden door een eenvoudig "retry y/n?" mechanisme opgevangen. De eerste crash moet ik nog tegenkomen.

Samengevat kan men zeggen, dat het gaat om een zeer professioneel afgewerkt produkt, waar elke ernstige programmeur veel plezier aan zal beleven. De hoge aanschafprijs is dus wel in alle opzichten gerechtvaardigd.

uit kilobytje



Some icing on the cake

uit kilobytje

Af en toe komt er op de QL-markt eens iets uit, dat te klein is om er een heuse review aan te wijden, maar niettemin interessant genoeg om het een blik te gunnen. Hier volgen enkele van deze dingetjes.

Tony Tebby's venerabele TOOLKIT II behoeft hier nog nauwelijks voorgesteld te worden. Een nadeel voor sommige doleinden is natuurlijk dan het hele monster in EPROM zit. Wat als je een programma wil maken, dat op iedere QL draait, maar Toolkit-extensies gebruikt ?

Alternatief 1: de gebruikers Toolkit II te koop aanbieden tegen gunstprijs, hetzij legaal, hetzij illegaal.
Alternatief 2: een image nemen van de ROM, met een disassembler het juiste entrypoint voor de call zoeken en het hele monstrum aldus in de boot opnemen. Ten eerste juridisch niet geheel proper , ten tweede nogal plaatsrovend (zeker voor mensen met niet upgraded QLs). Alternatief drie: de hele Toolkit II via een $$asmb directief inlinken tijdens de compilatie met QLiberator. Deze methode is wettelijk vrij clean, gezien de koper de Toolkit dan niet meer apart kan gebruiken. Evenwel, ieder programma krijgt dan een eigenlijk overbodige 16 K erbij, terwijl je meestal toch maar een routine of vier, vijf per programma gebruikt.

De RECONFIGURABLE TOOLKIT biedt nu een andere oplossing: met een speciaal menugestuurd configuratieprogrammaatje kunt u precies dat stel routines eruit trekken, dat u nodig hebt. Het hele ding is zelfdocumenterend: zelfs krijgt u, als u met de cursor op het veld van een routine gaat staan, een korte beschrijving ervan in een apart window.
De routines zelf zijn die uit de laatste nieuwe versie, 2.11, die u ook in de Trump Card vindt. In tegenstelling tot bij de EPROM-versie, werd ook de MG-patch (een klein stukje publicdomain machinecode, dat een nogal irritante bug in de grafische routines van de MG-rom wegwerkt) opgenomen: deze bevindt zich nu trouwens in de clubbliotheek voor wie er iets aan heeft. Als gebruikers van QL versies AH,JM of JS zich een toolkit willen maken, selecteren ze dat item gewoon niet.

En ziezo: als u nu een geweldig prgrammaatje geschreven hebt, dat bv. voor error trapping op de filebehandeling de Toolkit-functies FOP_IN,FOP_NEW,... gebruikt, dan kunt u gewoon even een toolkitje configureren met alleen die commando's (dat dan nog geen 2 K lang is), en het vervolgens met een $$asmb-directief inlinken tijdens de compilatie met QLiberator.

Van QLiberator is trouwens ondertussen de versie 3.1 uit, met de volgende belangrijke verbeteringen:
-hij draait onder QRAM, en maakt gebruik van de pointer-interface
-grote programma's kunnen overlayed worden
-sets routines,met QLiberator gecompileerd, kunnen gebruikt worden als Superbasic-extensie !!!! Dit zonder de restricties die een analoge jezuietentruc bij TURBO stelt. Zelfs kunnen EPROM-images, met header en al, van meerdere programma's (zelfs BASIC, machinecode,... door elkaar) worden geprepareerd. Eindelijk de simpele oplossing voor wie zes toolkits tegelijk in EPROM wil schieten
-integer for loops worden gesupporteerd d.m.v. een soort IMPLICIT typing
-communicatie van variabelen tussen tasks (GLOBAL,EXTERNAL,...) wordt nu ook gesupporteerd.

Uiteraard wordt de snelheid van Turbo in geen enkel opzicht ook maar benaderd: het is en blijft immers een pseudocompiler, ongeveer zoals Turbo-Pascal op de MS-DOS machines.

Tesamen met QLiberator werd ook QLOAD/QREF aangekocht. Voor het zeer luttele prijsje (£ 9.95) is dit produkt zijn geld beslist waard QLOAD, QSAVE en QLRUN zijn extensies die toelaten een Basic-programma in zijn inwendige vorm, i.p.v. in tekstvorm, te laden en te saven. Het nut hiervan zit hem in de explosieve tijdwinst bij grote programma's: immers, bij ht laden van een Basic-programma op de normale wijze wordt zeer veel tijd gebruikt bij de tokenizing (opnieuw opbouwen van de symbooltabellen etc...). Een programma van 60 K in Basic kan gemakkelijk bijna 10 minuten in beslag nemen om te laden (vanaf disk dan nog !): met QLOAD (na natuurlijk QSAVE i.p.v. SAVE gebruikt te hebben) wordt dit gereduceerd tot ca. 15 seconden !!!
Anders dan gelijkaardige systemen (FLRUN e.d.) is er geen probleem met afhankelijkheid van type QL en configuratie. Wel moeten, als er extensies worden gebruikt in het programma, deze zowel bij de QSAVE als bij de QLOAD geladen zijn, dit om gibberish in de interpreter-tabellen te vermijden. Als het dat maar is...

QFIND laat toe, een variabele of functie/procedure te zoeken in het ingeladen programma. Zeer handig in conjunctie met de full screen-editor uit TOOLKIT II. Search repeats zijn ook mogelijk.

QREF en al zijn varianten, tenslotte, laat toe een cross-referencing te doen van variabelen, functies, procedures,... eventueel met een wildcardstring. Dit voegt een faciliteit toe aan Superbasic, die door de wol geverfde programmeurs wel van de compilers op hun mainframe zullen kennen, maar zeker niet van een interpreter. Soms, als de boel om onverklaarbare redenen maar niet wil lopen, blijkt uit de xref dat een bepaalde variabele op een obscure plaats verkeerd gespeld werd...

Enfin, u merkt het, er zit nog steeds beweging in het QL-front. Wij houden u met veel genoegen op de hoogte..

Vraag en Aanbod

Te Koop Aangeboden:
Cumana dubbele 3.5 diskdrive 720KB in kast met ingebouwde voeding..... ƒ 425,--

CST diskinterface..... ƒ 100,--

Microvitec Kleuren-monitor..... ƒ 400,--

Epromkaart met doorvoerconnector en 128 KB programma's op eprom: Turbo-toolkit II, Copymachine, Qflash, Superbasic Extensions, Utility's van Qram, Qlone, Ultradisk, Graphic Constuction Kit..... ƒ 150,--

Speedscreen in Romkaart (voor achterin)..... ƒ 40,--

ICE in Romkaart (voor achterin)..... ƒ 40,--

QL-JS versie, 512KB (intern)..... ƒ 300,--

36 Cartridges per stuk..... ƒ 3,--

Reacties naar:
Bert Linschoten
Weidonklaan 2
s'Hertogenbosch
Tel. 073 - 214278

Te Koop Aangeboden:

2 diskdrives, 3 inch met 8 floppies, de drives zijn enkelzijdig, de floppies kunnen echter aan 2 zijden bescheven worden (totaal 360KB)
Vraagprijs..... ƒ 295,--
Inl. Hans Brinkmann
Tel. 070 - 456510

Te Koop

QL (512 intern) met Schön toetsenbord, dubbele diskdrive (3,5" 720KB elk) met interface, Ql-printer, Sciento monitor (amber), 40 cartridges met veel software, alles in één koop..... ƒ 1600,--
inl. W.J. van Duijnhoven
Tel. 04929 - 64937

Te Koop Aangeboden:

QL-JM, 128 K prijs..... ƒ 275,--

Fred Vink
Tel. 02230 - 342050
(tussen 18.00 en 22.00 uur).

Te Koop Aangeboden:

Interne geheugenuitbreiding (incl. documentatie) naar 640KB, enig solderen vereist.
Spectrum+ (48K) incl. cassette-recorder, 2 boeken, 3 tijdschriften, 9 originele spelporgramma's en orginele assembler. Eventueel ruilen tegen 3,5" 720K diskdrives.

Inl. Erwin Bolwidt,
Ouderkerk a/d Amstel
Tel. 02963 - 1564 (na half zeven)

Te Koop Aangeboden:

Sinclair QL-JS(128K), QL-printer, CUB kleurenmonitor, diverse handboeken en tijdschriften, 50 cartridges, centronicsinterface en diverse programma's (o.a. chess).
Vraagprijs..... ƒ 1000,--

Ad Dekkers
Rijksweg 122
6581 ES Malden
prive: 080 - 583487
werk: 040 - 605215

Te Koop Aangeboden:

In goede staat verkerende QL(JM) bijbehorende printer, ±50 cartridges veel programma's en veel documentatie.
Alles in één koop.
Vraagprijs..... ƒ 700,--

Martin de Zeeuw
Tel. 070 - 600162
(tussen 18.00 en 22.00 uur)

Te Koop Aangeboden:

QUB Kleuren monitor..... ƒ 400,--
2 x 3,5 inch Diskdrive (2 x 720k) Cumana ..... ƒ 425,--
128k Eprom kaart incl. 4 x 32k Eproms met:
- Toolkit II
- Toolbox
- GCK
- Qload Qref
- Copier
- Qram
- Super Basic Extensies
- Qflash

Dit is de inhoud van de 4 x 32k eproms, dus de Eprom kaart met 4 32k Eproms..... ƒ 125,--
Cartridges nog 8 setjes van 4 st. per set..... ƒ 15,--
Cartridges nog 6 setjes van 4 st. per set..... ƒ 20,--
Chess orgineel ..... ƒ 25,--
ICE op Eprom nog 1 stuks..... ƒ 40,--
Toolkit II op Eprom nog 2 stuks..... ƒ 40,--
Speedscreen op Eprom nog 1 stuks..... ƒ 40,--

Fred van der Neut
01807-10553 (alleen woensdag avond tot 22.00 uur)
010-4546372 overdag

VRAAG???

Heeft iemand een programma, of kan iemand een programma maken, waarmee een toets van het toetsenbord veranderd kan worden? Dus bijvoorbeeld om de min (-) te veranderen in de underscore (_) en andersom. Het gebruik spreekt voor zichzelf. Het liefst zodanig dat het in een bootfile opgenomen kan worden.

Oplossingen via:
Dirk de Vogel
Ireneweg 1
6862 HL Oosterbeek

of via het redactieadres:
Gerard van Rooijen
Gruttostraat 15
3435 DJ Nieuwgein

---

4 juni QL-dag in Utrecht

Zaterdag 4 juni 1988
QL- bijeenkomst in de
ir. A.J. Versfelt MTS
Grebbeberglaan 15 te Utrecht